UNIVERSIDAD CARLOS III DE MADRID

Proyecto fin de carrera

MEJORA DE UN SISTEMA DE ILUMINACION MODULAR BASADO EN LED DE ALTO BRILLO (HBLED)

Autor: David Domínguez Moreno

Tutor: Pablo Zumel Vaquero

Titulación: I.T.I. Electrónica

Agradecimientos

En este proyecto de fin de carrera, quería agradecer en especial a mis padres y a mi hermana por darme ánimos, durante todos estos años de la carrera, y aguantarme en esos momentos de estrés y mal humor.

Este Proyecto va para mis abuelos, que siempre han estado hay durante toda la carrera apoyándome e interesándose por como evolucionaba en la carrera, y en especial a mi abuelo que falleció este año.

También agradecer a mi tutor, gracias a él este proyecto ha sido posible, ya que durante un año y medio, he aprendido muchas cosas que me han hecho crecer como ingeniero.

Por último se lo dedico a todos mis compañeros que he conocido durante toda la carrera, desde el primer año hasta el último, que hemos estado estudiando codo con codo.

¡¡¡¡GRACIAS A TODOS!!!!

DESCRIPCION DEL PROYECTO

El proyecto consiste en la mejora del funcionamiento de un sistema de iluminación mediante led de alto brillo, basado en el encendido y apagado de los led de 5 módulos, sincronizados con la red, para conseguir conformar la corriente inyectada a la red sin necesidad de almacenamiento de energía. Se mejora la forma de onda de la corriente mediante la modulación PWM del encendido y apagado de los módulos led.

Los ángulos de conmutación de cada módulo, que nos dirá el instante de encendido y apagado de los led, está basado en un estudio realizado en MATLAB.

Para la sincronización, se ha realizado un circuito y mediante la programación de la FPGA, con un programa en VHDL, se ha conseguido que el encendido y apagado de los led, estén sincronizados con la señal de red.

Por último, las señales de salida de la FPGA, se pasaran por un convertidor-reductor, realizado en otro proyecto, para que la corriente que circule por los led sea constante.

1 Contenido

2 Tabla de ilustraciones

1. INTRODUCCION

1.1 MOTIVACION

Este Proyecto de Fin de Carrera ha sido una gran motivación para mí, ya que se han utilizado varios campos de la electrónica, que me han servido para recordar y ampliar mis conocimientos.

El objetivo principal de este proyecto es mejorar un sistema de iluminación conectado a la red convencional de corriente alterna basado en la conexión por módulos de diodos de iluminación HBLED sin condensador de almacenamiento de energía. Cada módulo de HBLED está alimentado con una corriente constante mientras está conectado a la red. La corriente consumida por la carga, que son los módulos HBLED, debe emular una onda sinusoidal mediante la conexión y desconexión de los módulos HBLED a lo largo del ciclo de tensión de red. En el sistema se debe evitar utilizar condensadores electrolíticos para alargar el tiempo de vida del sistema de iluminación. Además se deben mejorar los saltos de conexión cada módulo en la onda sinusoidal de corriente de entrada, para que no se produzcan de una forma brusca, con el fin de mejorar las prestaciones de una instalación lumínica de led ya existente.

Para conseguir el objetivo de mejorar el sistema de iluminación, el proyecto se ha dividido en diferentes partes.

Por una parte, se dividirán los led, del tubo del que disponemos, para 5 módulos. Se realizaran las correspondientes conexiones, con un cable a la entrada de cada módulo y otro a la salida, para su correspondiente alimentación.

Por otra parte, se realizara en MATLAB; primero un diagrama de bloques en SIMULINK, con la intención de obtener una secuencia, de instantes de conmutación de cada módulo y posteriormente en MATLAB, mediante un código, obtener los datos de esos instantes de conmutación.

Por otra parte, se realiza una placa, con el objetivo, de sincronizar la señal de salida con la red, mediante el filtro UAF42.

Por último, se realizara un código en VHDL, se introducirá en la memoria interna de la FPGA, para sincronizar, nuestra señal de salida del circuito, con los instantes de conmutación.

Las señales de salida de la FPGA se utilizarán para la conexión y desconexión de un convertidor-reductor controlado en corriente, que consigue una corriente constante por cada módulo mientras está encendido. La combinación de todos los módulos producirá una corriente de red, emulando una onda sinusoidal. El convertidor-reductor utilizado se ha desarrollado en un proyecto realizado anteriormente.

1.2 PLANTEAMIENTO DEL PROBLEMA

Para conseguir esa corriente de red, emulando una onda sinusoidal, se realiza el siguiente planteamiento:

HBLED, son leds de alto brillo, que son utilizados para la retroiluminación de pantallas, iluminación en coches, luces de emergencia, comerciales, residenciales e iluminación de calles, debido a su alto tiempo de vida. Ya que una gran ventaja de este tipo de led es su elevada duración en el tiempo, se debe evitar tecnologías como la de los condensadores electrolíticos en el acondicionamiento de estos led, ya que perjudica su alto tiempo de vida.

Los led se alimentan con corriente continua, y nuestra señal de red, es alterna, por lo que se tiene que transformar la señal. Ya que una de las mayores ventajas de los led de alto brillo, es su tiempo de vida, se tiene que realizar el circuito electrónico de acondicionamiento, sin condensadores electrolíticos, ya que estos reducen el tiempo de vida del circuito, como se puede observar en la figura 1.1:

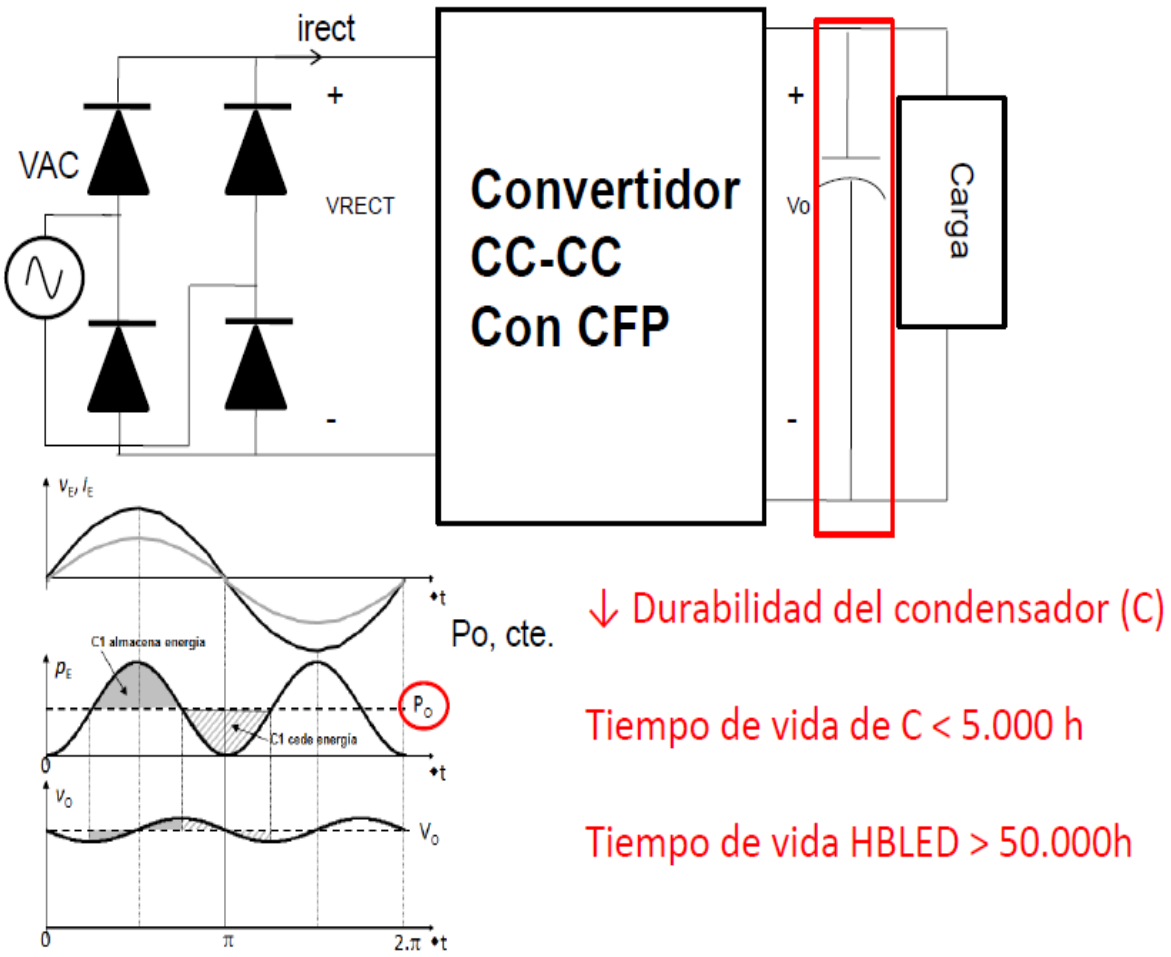


Figura 1.1: Potencia consumida por la carga, mediante un condensador electrolítico. Ref "Estrategia de control de HBLED con reducción del condensador de almacenamiento basada en la modularización de la carga. Pablo Zumel"

Se observa en la figura 1.1 como la potencia consumida por la carga es constante, que es lo que se busca para alimentar a los led. Hay que intentar adaptar la señal consumida por la carga, a la potencia que la red nos genera para que la corriente demandada de la red sea lo más sinusoidal posible y evitar la perturbación de otros equipos por efecto de la inyección de armónicos de corriente en la red. Una de las soluciones es utilizar un condensador, como se ve en la figura 1.1, que almacene energía en los momentos en los que la red no puede dar la

potencia que demanda la carga porque tensión y corriente son cero, o tienen un valor reducido. Este condensador debido a sus valores de capacidad y tensión suele ser electrolítico. Pero como ya se ha dicho anteriormente, hay que buscar otra solución, sin que se tenga que utilizar el condensador electrolítico, ya que perjudica al tiempo de vida de la instalación.

Para resolver esto, se ha divido la carga en varios módulos, con el fin de que funcione, cada uno de ellos, de una forma espaciada en el tiempo. Por lo que no es necesario almacenar energía, como en el caso del condensador. Esta solución puede observarse en la figura 1.2:

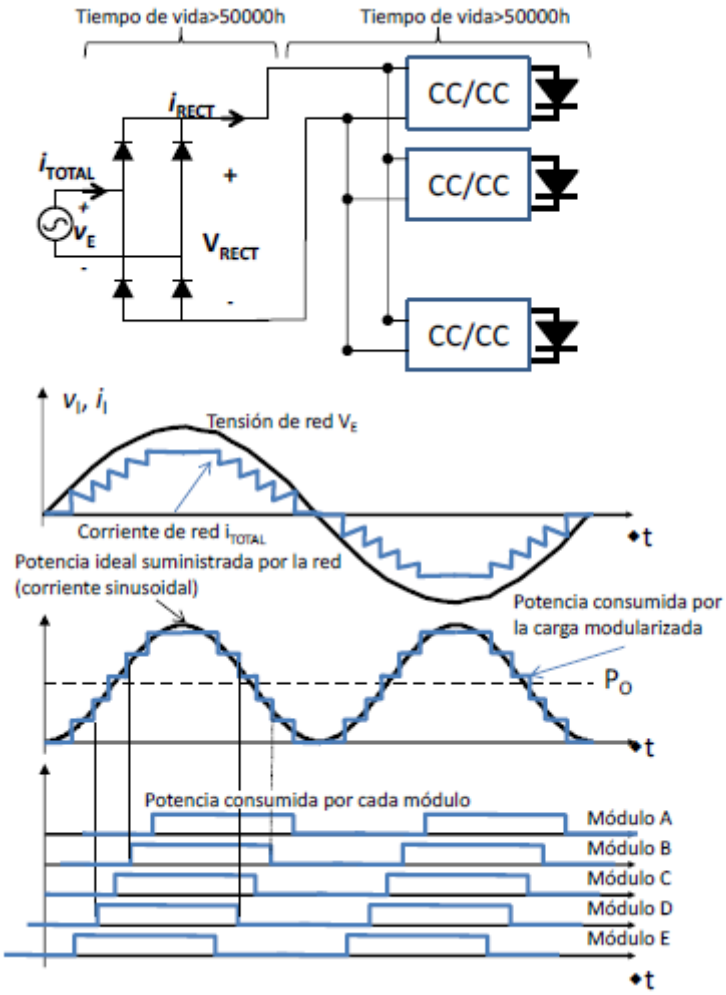


Figura 1.2: Potencia consumida por la carga, mediante la modularización de la señal de red. Ref:"Estrategia de control del condensador de almacenamiento basada en la modularización de la carga .Pablo Zumel"

Se observa que cada módulo, cuando se activa, consume una potencia constante. Por lo que la potencia total consumida por la carga, es la suma de la potencia consumida de cada módulo. Controlando la encendido y apagado de los módulos, se podrá conseguir una corriente de entrada "sinusoidal".

En este caso, se ha optado por la configuración en paralelo de los módulos. Se requiere una menor potencia de los convertidores, con una señal de control, para activar y desactivar cada módulo. La configuración en serie, se puede usar solo un convertidor cc-cc, pero se necesitan interruptores para conectar o desconectar cada módulo.

1.3 OBJETIVOS

1) Configuración del tubo de led, del que se dispone, se calcula para que todos los módulos tengan el mismo y el máximo número de led posibles, dejando la separación correspondiente entre ellos para la alimentación de cada módulo. El resultado final es de 5 módulos, de 24 led cada módulo.

2) Sincronización con la red mediante filtro paso banda de 2º orden, con elevada Q (UAF42).

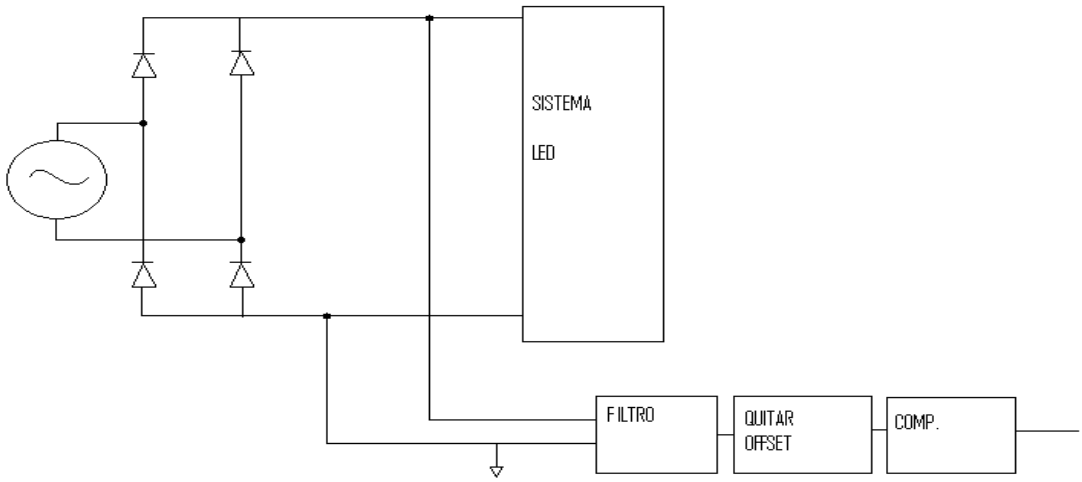


Figura 1.3 Diagrama de bloques de sincronización con la red

3) Generador de pulsos:
-Hacer que funcione en SIMULINK.
-Obtener unos vectores con los instantes de conmutación de cada módulo.
-Estudiar los armónicos de la onda resultante en función de la modulación PWM.

4) Implementarlo en el sistema con FPGA que existe y caracterizarlo.

5) Comprobar los resultados y sacar conclusiones.

Se muestra a continuación, los distintos bloques, en los que se ha dividido el proyecto para alcanzar nuestro objetivo:

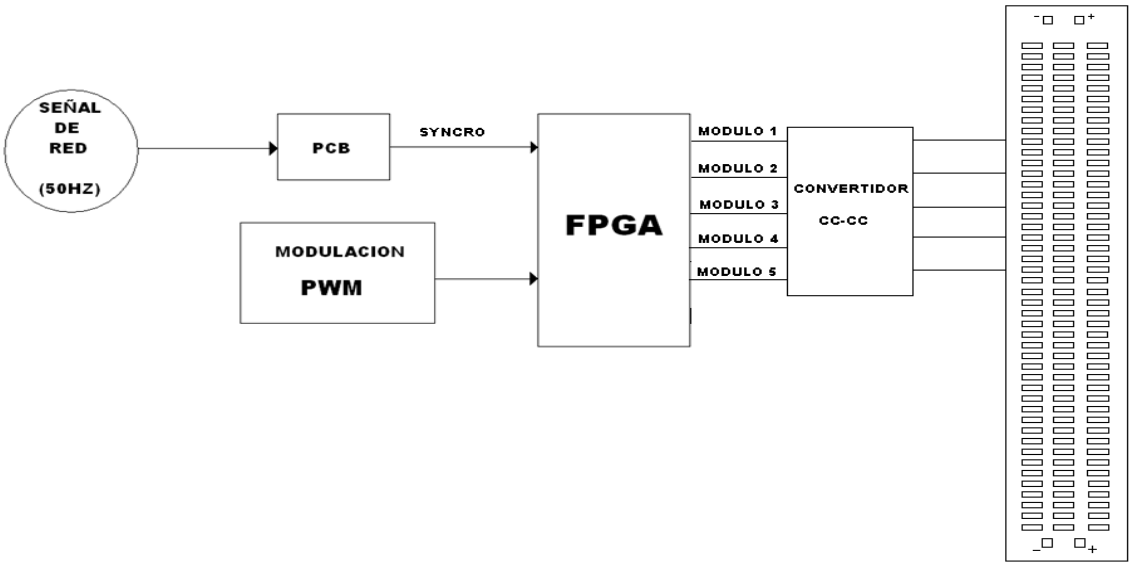


Figura 1.4 Diagrama de bloques de cada una de las partes del proyecto

Se alimenta nuestra PCB, con la señal de red de 50(Hz), con una tensión menor inicialmente, de unos 45(V), para realizar las pruebas necesarias, antes de alimentarla con una fuente mayor, que llegara hasta unos 150(V).

Se necesita una señal que esté sincronizada con nuestra señal de red, por lo que se utiliza el filtro UAF42, el cual nos genera 3 señales. Se elegirá aquella señal, que su paso por cero coincida con el de la señal de red. Se utilizara el comparador del UAF42, ya que se necesita una onda cuadrada. La señal syncro, es la señal de salida de la PCB.

La modulación PWM, se realizara en MATLAB, para calcular los instantes de conmutación de cada módulo.

En la FPGA, se sincronizara ambas señales, la señal “syncro”, con los instantes de conmutación mediante un código en VHDL. Posteriormente pasaremos las señales por un convertidor-reductor, realizado en otro proyecto, antes de mandar las señales al tubo de led.

1.4 HERRAMIENTAS UTILIZADAS

En la realización de nuestro Proyecto de Fin de Carrera se han utilizado varias herramientas:

-MATLAB (se consiguen los instantes de conmutación de cada módulo)

-ORCAD (se consigue que nuestra placa tenga el correcto funcionamiento)

-XILINX (se consigue, mediante un código en VHDL, que este sincronizado la salida de nuestro circuito con los instantes de conmutación)

-ADEPT (se utiliza para mandar el programa creado en XILINX, del PC a la FPGA)

2 TUBO DE LED

2.1 HBLED

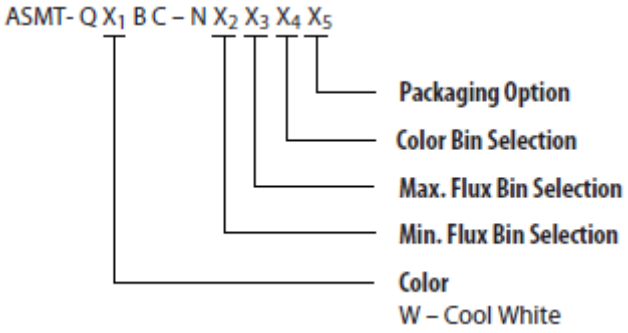
En este apartado, se explica el tipo de led utilizado y sus características. Y la distribución que se ha realizado en el tubo de led, para dividir el total de los led, entre los 5 módulos, de los que va a constar el proyecto. Además se explica la correspondiente alimentación de cada led y su polaridad.

2.1.1) Definición y modelo

El modelo del led que se ha utilizado en este proyecto es: *ASMT-QWBC-NHJOE*. Su uso generalmente, está vinculado a aplicaciones en automoción, y en signos y señales, en electrónica.

2.1.2) Características

"ASMT-QWBC-NHJOE", este tipo de led proviene de la familia ASMT-QxBC -Nxxx, se indica el significado de cada letra y sus características:



Device Selection Guide ($T_J = 25°C$)

		Luminous Flux, Φ_V[1] (lm)				
Color	Part Number	Min. Flux (lm)	Typ. Flux (lm)	Max. Flux (lm)	Test Current (mA)	Dice Technology
Cool White	ASMT-QWBC-NHJ0E	25.5	30.0	43.0	150	InGaN
	ASMT-QWBC-NJK0E	33.0	38.0	56.0	150	InGaN

Figura 2.1: Modelo de hbled utilizado

La luz de un LED, es proporcional a la corriente que le atraviesa, y la temperatura, es un factor a tener en cuenta para que los led tengan un correcto funcionamiento. A continuación, se muestra cómo reaccionan este tipo de led, antes estos parámetros.

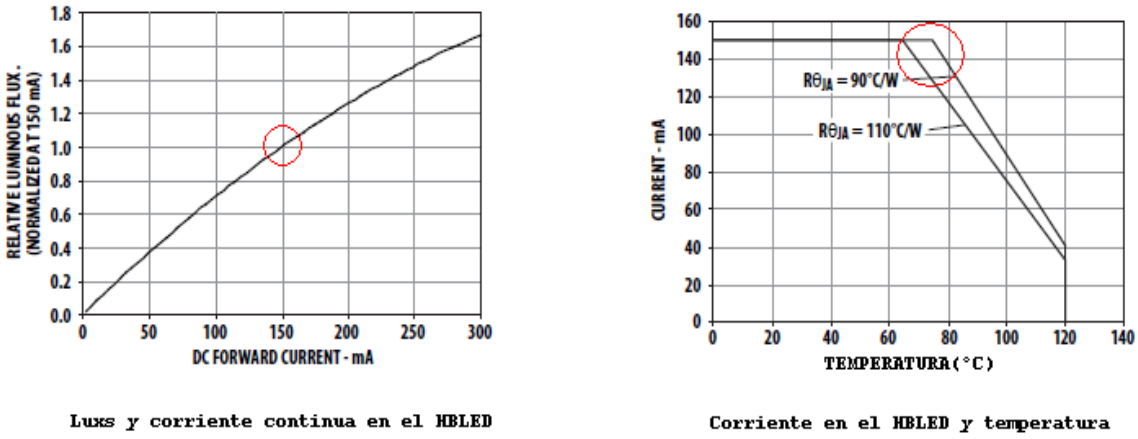


Figura 2.2: Características de hbled

A medida que aumenta la corriente que pasa por los led, la luz que proporciona es mayor. A partir de 70˚C, aproximadamente, la corriente por los led disminuye, así que al superar esta temperatura, los led no se iluminaran correctamente. La temperatura máxima que soporta este tipo de led son 125ºC, como se observa en la figura 2.2.

Estos led poseen una gran eficacia lumínica y un gran tiempo de vida, con respecto a otros instrumentos lumínicos, se comprueba en estas graficas:

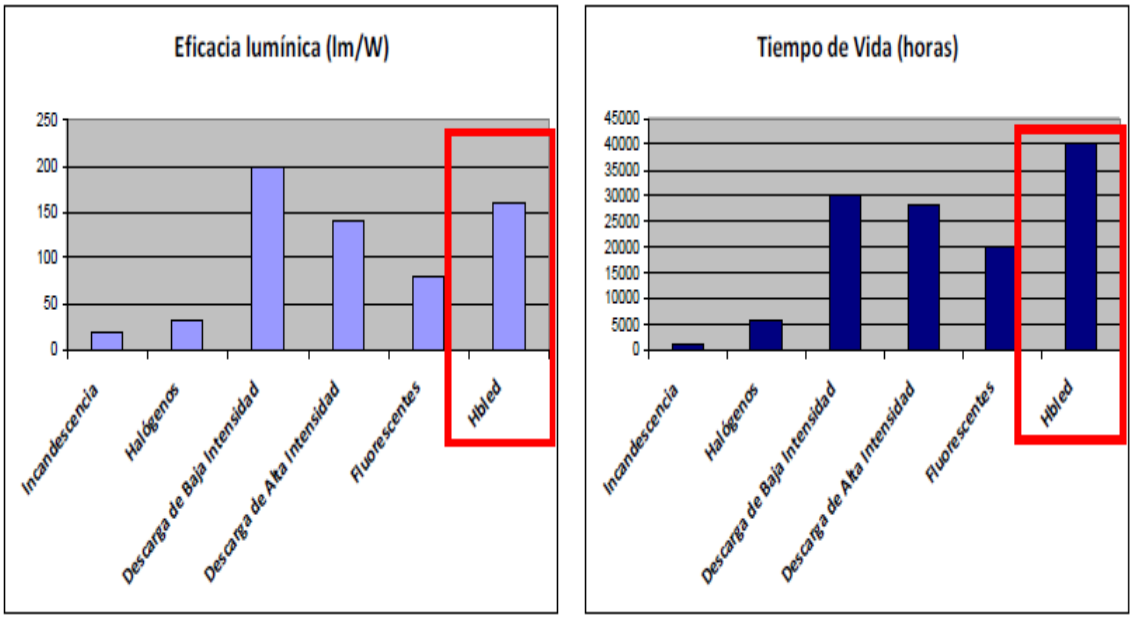


Figura 2.3: Eficacia lumínica y tiempo de vida de los hbled

2.2) DISTRIBUCION DE LOS LED

Se dispone de un tubo de led, de 3 columnas y 46 filas, que hacen un total de 138 led.

Tras un estudio, en el que se ha tenido que dividir, todos los led entre los 5 módulos y la alimentación correspondiente de cada uno de los módulos, y que cada módulo tiene que tener el mismo número de led; la distribución del tubo de led quedaría de la siguiente forma:

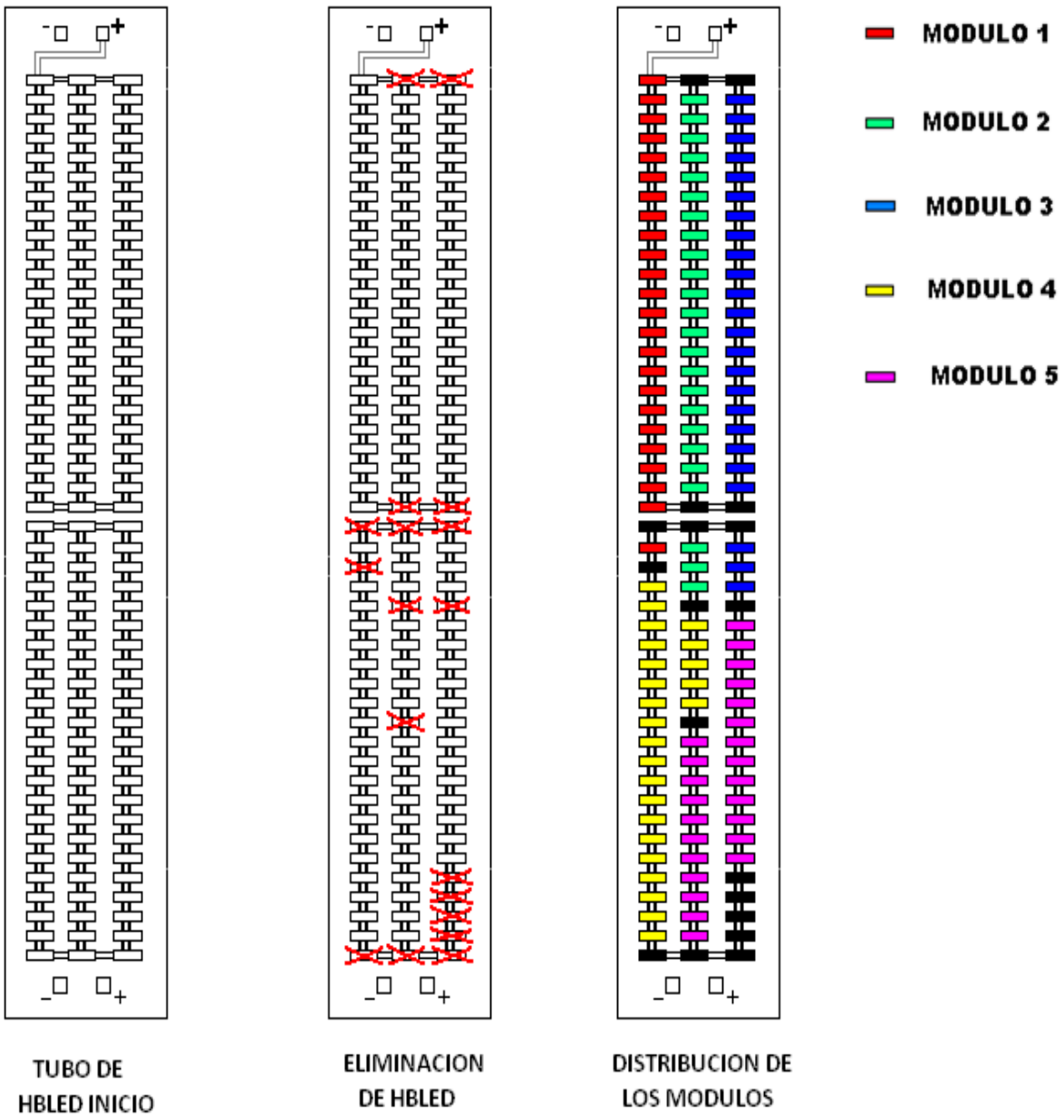


Figura 2.4: Distribución de cada módulo en el tubo de led

El tubo de led, del que se disponía, constaba de 2 bloques, y en cada bloque constaba de 3 filas de led en serie, por lo que para, dividir todos estos led en 5 módulos, y que los led de cada módulo este en paralelo, se realizó un estudio en el que se tenía que quitar algunos led, como se observa en la figura 2.4.

Finalmente, sé queda distribuido el tubo de led de la siguiente forma:

5 módulos con 24 led cada modulo

Se utilizan 120 led, de los 138 led que se tenían de inicio, ya que se han tenido que hacer las eliminaciones de algunos led, como se ha explicado anteriormente.

Se muestra la polaridad de los led, y las conexiones entre ellos. Y como se alimenta cada módulo, con un cable (+) a la entrada de cada módulo y con un cable (-) a la salida de cada módulo.

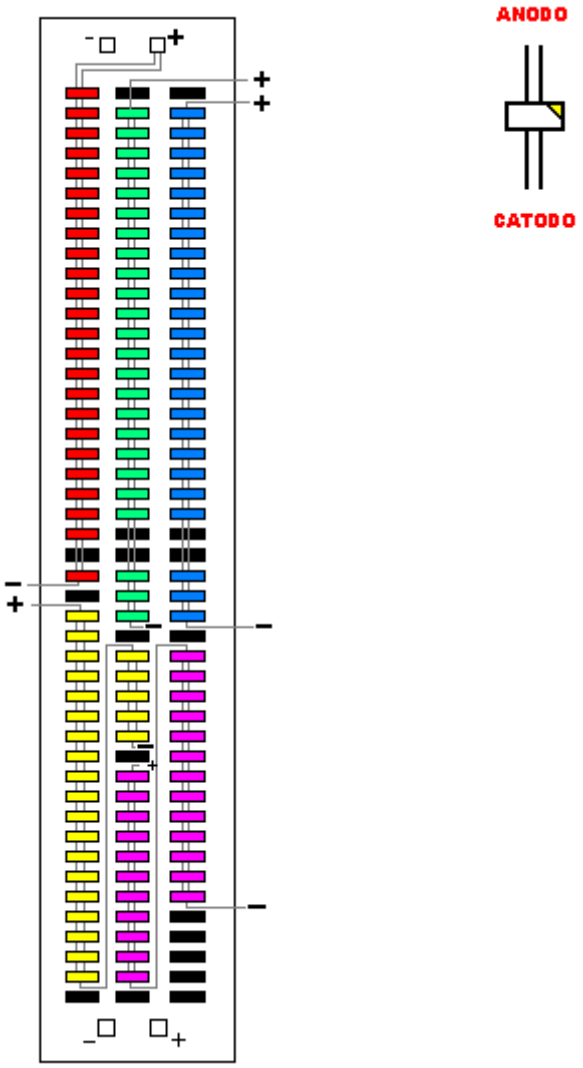


Figura 2.5: Conexiones de los hbled

Los led o diodo emisor de luz, es un semiconductor unido a dos terminales: ánodo y cátodo, positivo y negativo respectivamente, como se puede observar en la figura 2.5. Los led son dispositivos electrónicos semiconductores, funcionan con corriente continua, tienen polaridad, por lo que es imprescindible, para su correcto funcionamiento, que sean conectados en el sentido correcto, es decir, la corriente entra por el terminal positivo(ánodo) y sale por el negativo(cátodo), así, se iluminaran correctamente los led.

2.3) RESULTADO FINAL

El tubo de led y las conexiones necesarias de cada módulo, con el correspondiente cableado de cada uno de ellos, queda de la siguiente forma:

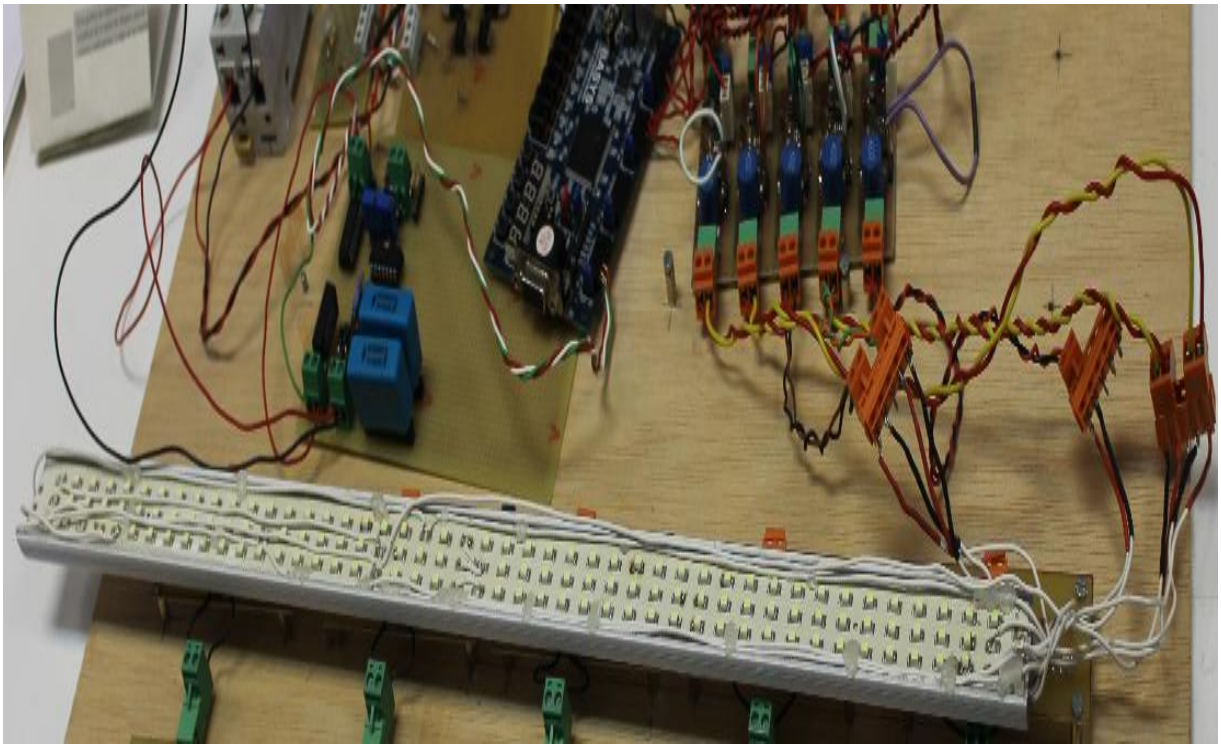


Figura 2.6: Foto del tubo de led

Se observa, como para cada módulo de led, se dispone de un convertidor-reductor. Necesario para la modulación de la señal de red y conseguir una potencia constante, en el sumatorio de las potencias consumidas por cada módulo.

3 GENERADOR DE PULSOS

3.1) ESQUEMATICO DEL CIRCUITO Y OBJETIVOS

El esquemático, de modulación PWM a varios niveles, que se ha a desarrollado, está realizado en SIMULINK, una aplicación de MATLAB.

OBJETIVOS

El objetivo de este bloque es calcular los instantes de conmutación de cada uno de nuestros módulos, los cuales, quedaran definidos en una tabla. Se calculara el instante de tiempo de encendido y apagado de los led de cada módulo, mediante una configuración de bloques realizada en MATLAB.

El periodo de tiempo en el que se realizan las simulaciones es de 10(ms), debido a que la señal de nuestro circuito, en la que se van a producir los ángulos de disparo, tiene una frecuencia de 100(Hz), como veremos en el siguiente capítulo .Con esto se consigue que nuestro rango de tiempo en donde se producen los instantes de conmutación, coincida con el periodo de nuestra señal de sincronismo de nuestro circuito.

El esquemático para conseguir nuestro objetivo es el que se observa en la figura 3.1, posteriormente explicaremos cada bloque, hasta conseguir la tabla con los instantes de conmutación de cada módulo.

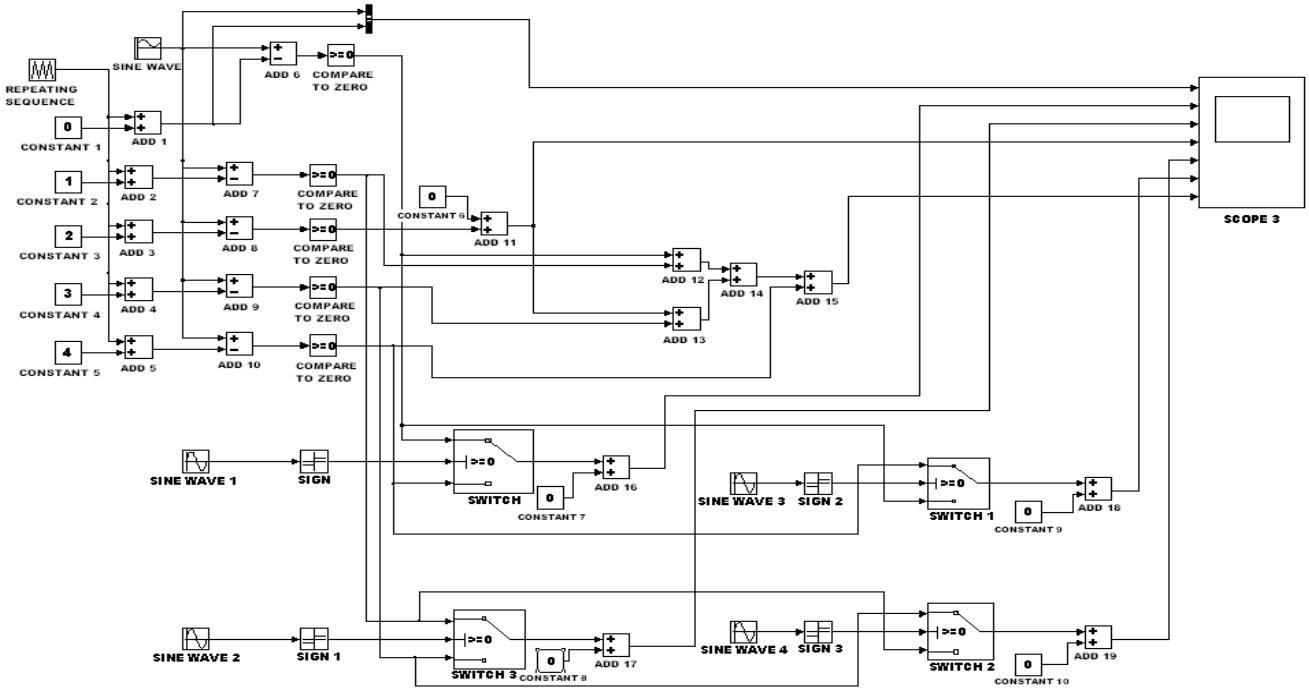


Figura 3.1: Esquemático de la modulación PWM

3.2) MODULACION PWM

3.2.1) Definición

PWM: MODULACION POR ANCHO DE PULSO (pulse-whist modulation)

La modulación por ancho de pulsos de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una sinusoidal o cuadrada), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

La generación de la PWM se realiza por modulación de una portadora, por medio de una señal de referencia o moduladora. La señal portadora es una triangular y la moduladora es una sinusoidal.

Esta técnica se controla por dos variables:

- Coeficiente m_f: controla el número de pulsos por semiperiodo.
  $m_f = \frac{frecuencia\_triangular}{frecuencia\_senoidal}$; Como se puede observar en la figura 3.2 .Si se varia el valor de la frecuencia de la portadora y se deja fijo el de la moduladora, se consigue aumentar el número de conmutaciones.

- $m_a = \frac{amplitud\_senoidal}{amplitud\_triangular}$; Si se varia el valor de la amplitud de la sinusoidal se consigue variar el ancho de los pulsos

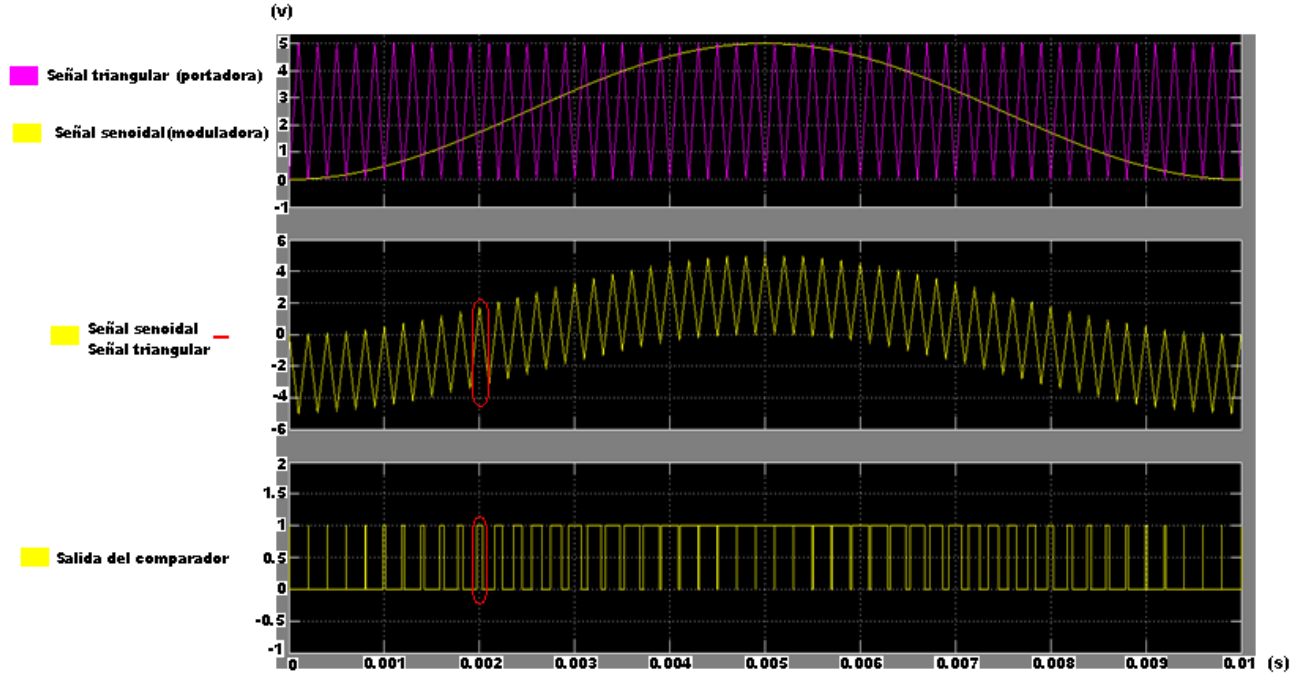


Figura 3.2: PWM de un solo nivel

Se observa en la figura 3.2: se resta la señal sinusoidal menos la señal triangular, obteniendo el resultado de la segunda grafica; pero lo que se quiere obtener es una sucesión de unos y ceros, que corresponde con el encendido y apagado de los led, para ello esta señal obtenida, la

pasamos por un comparador. La resta entre la señal sinusoidal y la señal triangular, cada vez que esta diferencia es mayor o igual que cero, la señal a la salida del comprador se pone a uno.

Mientras que cuando la diferencia de ambas señales es menor que cero, la señal a la salida del comparador se pone a cero.

Esta modulación PWM que se ha descrito es para un solo nivel, sin embargo, nuestro objetivo es conseguir los instantes de conmutación para 5 módulos. Hay realizar una modulación PWM para 5 niveles:

Siguiendo de ejemplo la PWM, que se ha descrito anteriormente. Lo primero que hemos pensado es como se puede distribuir los instantes de conmutación, de los 5 módulos, durante el periodo de los 10(ms). Para ello se comparan dos señales (una sinusoidal y una triangular).La señal sinusoidal se mantendrá fija, mientras las 5 señales triangulares irán incrementándose en uno su amplitud sucesivamente. Este incremento de amplitud en la señal triangular es necesario para que una vez que termine de parpadear el primer módulo, se producirá una comparación en un nivel superior, de la señal sinusoidal con la señal triangular (con amplitud uno) para conseguir la secuencia de disparo del segundo módulo. Así seguiremos comparando la señal sinusoidal con las señales triangulares hasta que se acabe el periodo de nuestra señal sinusoidal.

El ciclo de trabajo de una señal periódica, es el ancho relativo de su parte positiva en relación con el periodo. Expresado matemáticamente:

$$D = \frac{t}{T};$$

Dónde: D→ es el ciclo de trabajo

t→ es el tiempo en el que la función es positiva (ancho de pulso).

T→ es el periodo de la función

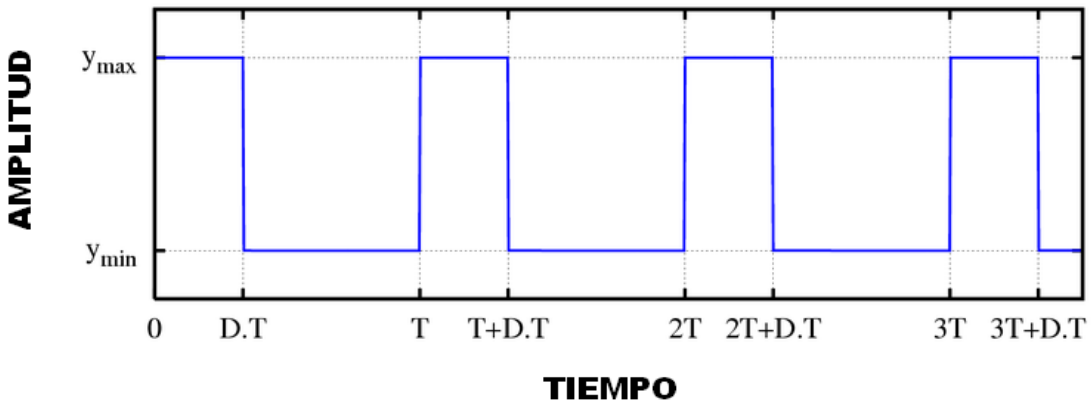


Figura 3.3: Ciclo de trabajo de una señal periódica

En nuestra modulación PWM, la señal generada tendrá frecuencia fija y tiempos de encendido y apagado variables. En otras palabras, el período de la señal se mantendrá constante, pero la cantidad de tiempo que se mantiene en alto y bajo dentro de un período puede variar.

En la actualidad existen muchos circuitos integrados en los que se implementa la modulación PWM, además de otros muy particulares para lograr circuitos funcionales que puedan controlar fuentes conmutadas, controles de motores y algunas otras aplicaciones.

3.2.2) Estudio de la fuente sinusoidal y triangular

Lo primero que se tiene que tener en cuenta para llevar a cabo la modulación PWM, a varios niveles, son las características de las dos fuentes.

La frecuencia de la señal moduladora o sinusoidal, es la que nos va a marcar el periodo en el que se va a producir los instantes de conmutación. Esta señal sinusoidal solo tiene componente positiva.

Las señales portadoras o triangulares, nos marcara el número de módulos que vamos a utilizar, y depende de su frecuencia que se tengan más o menos instantes de conmutación.

A continuación mostramos los valores de cada fuente:

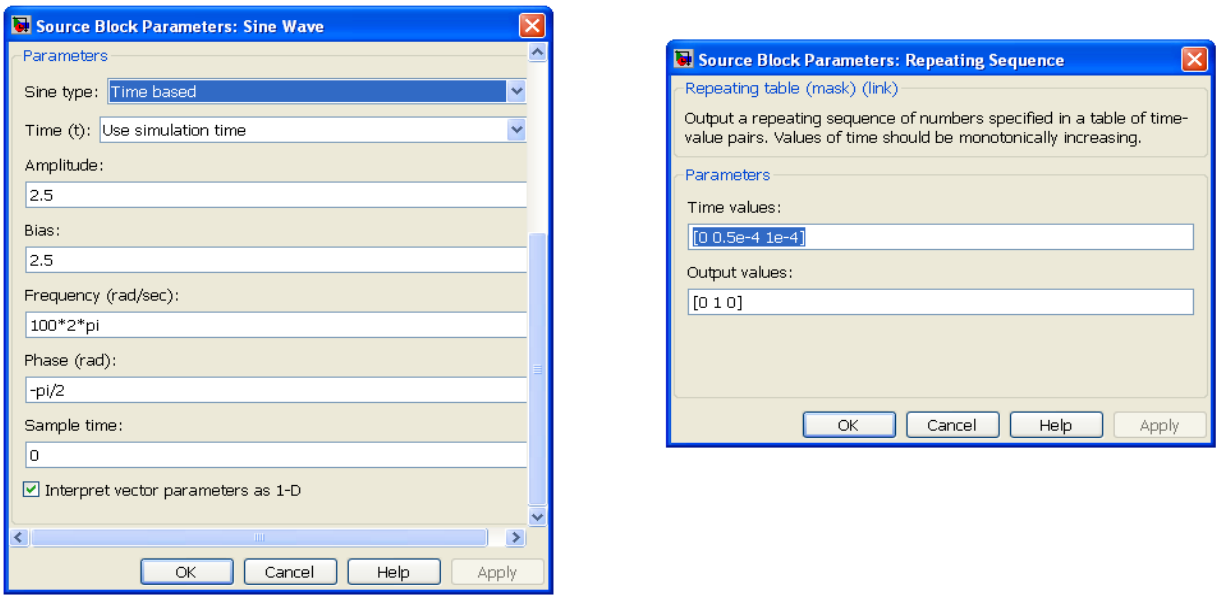


Figura 3.4: Parámetros de cada fuente

Vamos a ir explicando todos los pasos que hemos realizado en MATLAB hasta conseguir nuestro objetivo. Lo primero vamos a ver los parámetros de las dos señales utilizadas, que posteriormente vamos a ir comparando la señal sinusoidal con varias señales triangulares:

- SINE WAVE: onda sinusoidal. Su amplitud es de 2.5 (V).Se le suma 2.5 BIAS para trasladar la señal sinusoidal a la parte positiva. Frecuencia de 100 (Hz).

- REPEATING SEQUENCE: onda triangular. Su amplitud es de 1(V), para cada módulo le iremos sumando una constante(

). Periodo es de 0.1 (ms).

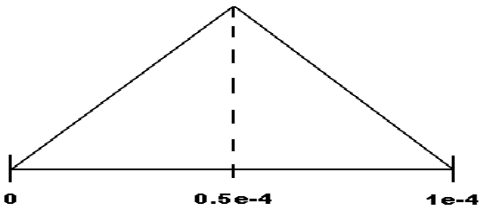
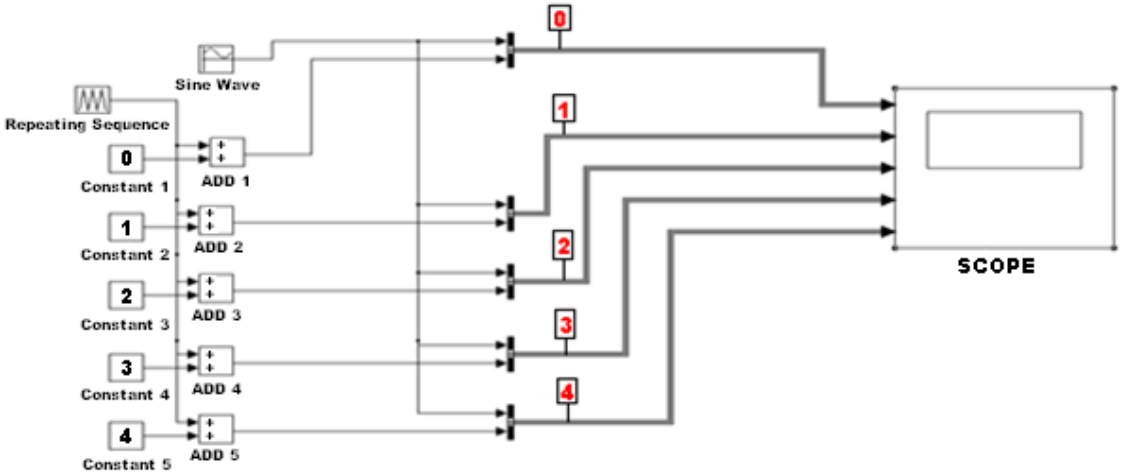


Figura 3.5: Representación de los valores de la señal triangular

El periodo de nuestra señal triangular es de 0.1 (ms), el cual se repetirá 100 veces durante el periodo de nuestra señal sinusoidal que dura 10(ms).Este periodo de la triangular se conservara para todos los módulos, lo único que vamos a ir variando va a ser su posición, ya que solo vamos a ir incrementando en 1 su posición, en el eje vertical, sumándole constantes a la de nuestro primer módulo. Vamos a ir explicando mediante unas graficas:



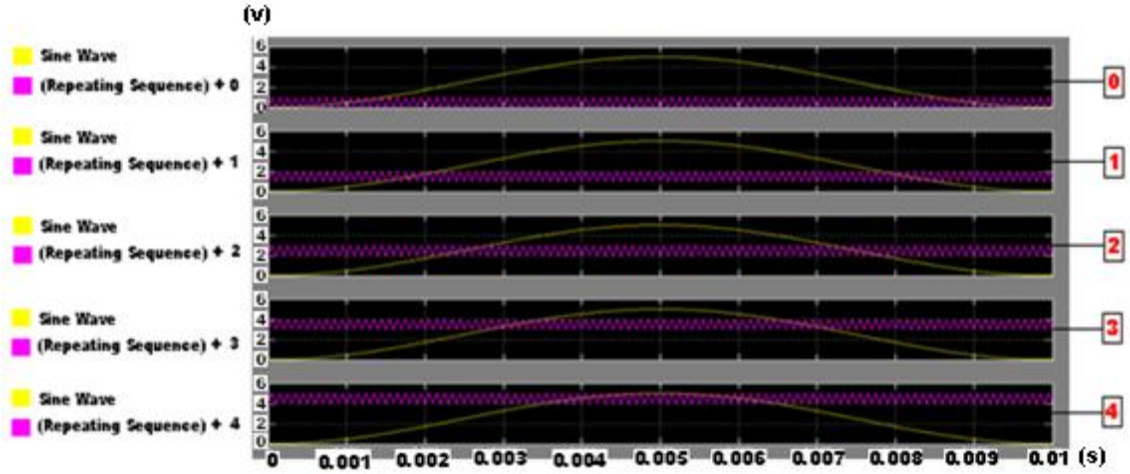


Figura 3.6: Las 2 fuentes: señal sinusoidal (moduladora) y las 5 triangulares(portadora)

Después de analizar las características, en general, de cada fuente. Se comparan ambas (señal sinusoidal menos la señal triangular), para que parte de la señal resultante, quede por encima y por debajo de cero, dando lugar a los instantes de conmutación de cada módulo, que

coincidirá con el parpadeo de los led. Posteriormente, se pasara esa señal por un comparador y se obtendrá una combinación de unos y ceros, que nos marcaran los instantes de conmutación de cada módulo.

El valor de la constante del primer módulo, tendrá un valor cero. Ya que es necesario que la señal portadora, coincida con el origen de la señal moduladora. Posteriormente, se incrementan en uno las constantes de los demás módulos, para que todos estén sincronizados correctamente, se vayan sucediendo unos módulos a otros, empezando la modulación al principio de la señal moduladora y acabando al final de la misma. Si la señal portadora del módulo 1, no comenzara en el origen, la portadora del módulo 5 quedaría por encima de la señal moduladora, sin llegar a producirse los ángulos de conmutación.

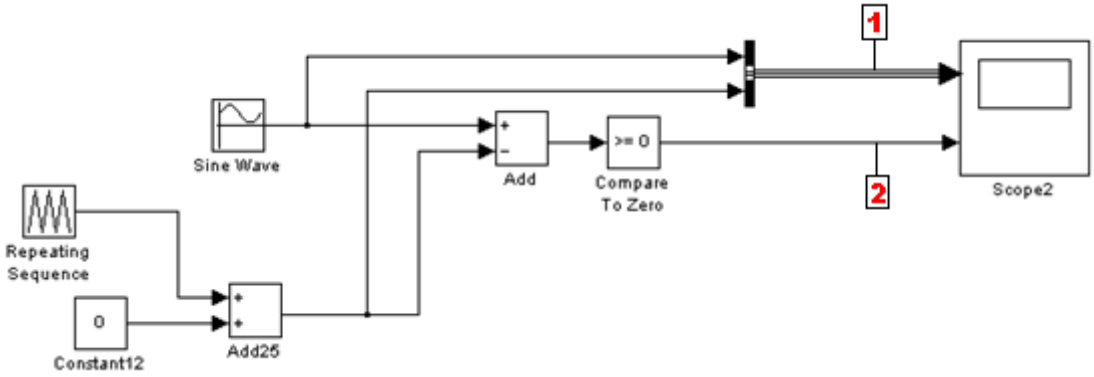


Figura 3.7 Esquemático de las fuentes utilizadas en MATLAB

Resultado del osciloscopio de la figura 3.7:

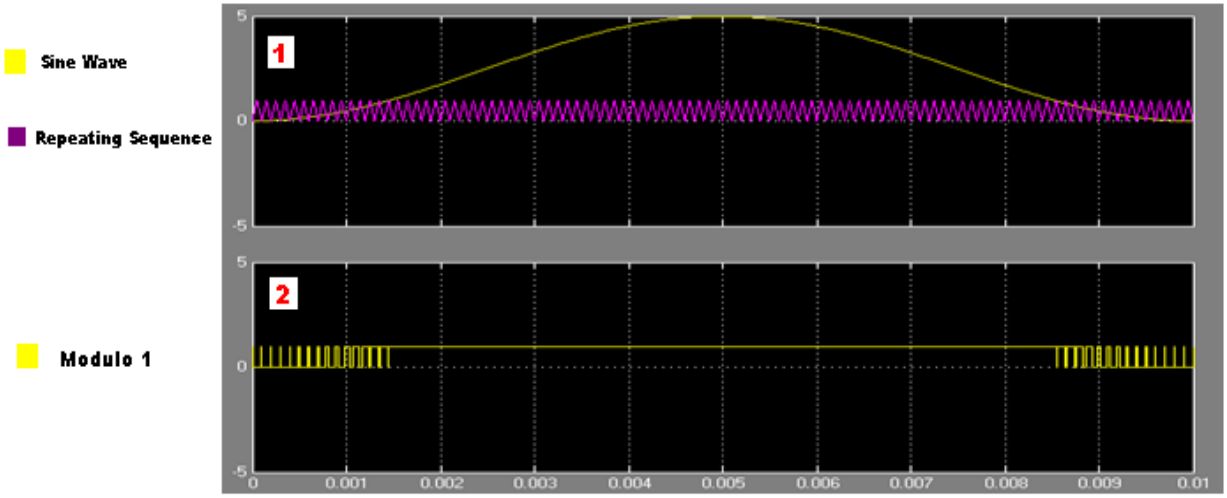


Figura 3.8: Simulación de la comparación de las 2 fuentes, para calcular los ángulos del modulo1

NOTA: Los parámetros generales para la realización de las simulaciones en SIMULINK, son los siguientes:

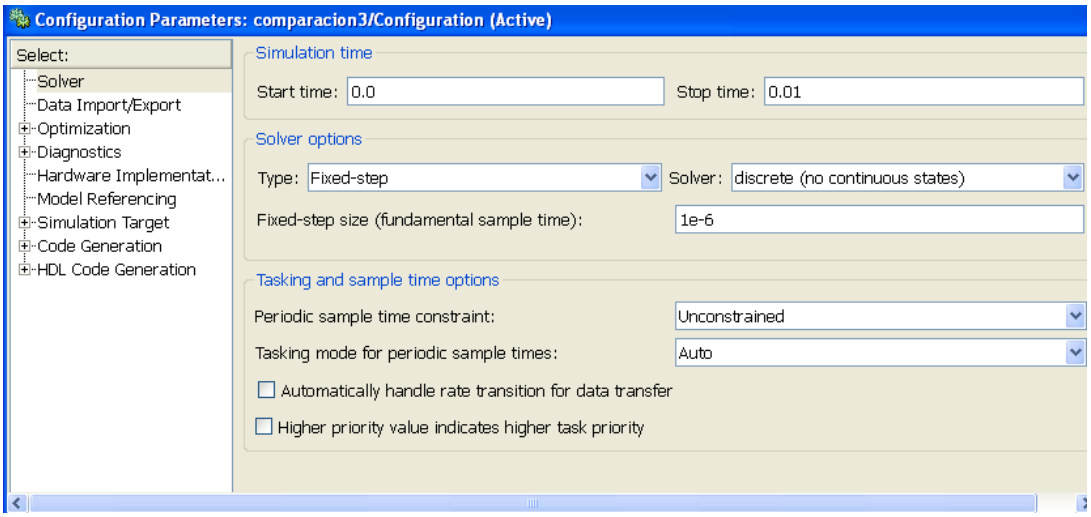


Figura 3.9: Parámetros de la simulación en MATLAB

→El periodo va desde 0 a 0.01(s)→10(ms), ya que es el periodo que se necesita, para que coincida con el periodo de la señal de salida de nuestra placa.

→Queremos que vaya leyendo datos cada 1(µs).Es decir, cada 1(µs), se obtendrá un valor (1 o 0), de la modulación PWM. Al final obtendremos una tabla, con 10.000 puntos (correspondiente a cada µs) de cada módulo y con su correspondiente valor.

A continuación, se explica los valores que deben de tener ambas señales para cada uno de los 5 módulos:

→1º modulo: onda sinusoidal y triangular con constante de 0(

).

→2º modulo: onda sinusoidal y triangular incrementando su posición en uno, ya que le hemos sumado una constante de valor 1(

), a nuestra señal triangular original.

→3ºmodulo: onda sinusoidal y triangular a la que incrementamos en 2 (

).

→4ºmodulo: onda sinusoidal y triangular a la que le incrementamos en 3(

).

→5ºmodulo: onda sinusoidal y triangular a la que le incrementamos en 4(

).

A continuación, se muestra gráficamente la comparación de ambas señales, con los valores de cada fuente, que anteriormente hemos descrito. A la señal sinusoidal le restamos las 5 señales triangulares correspondientes. Se muestra el esquemático:

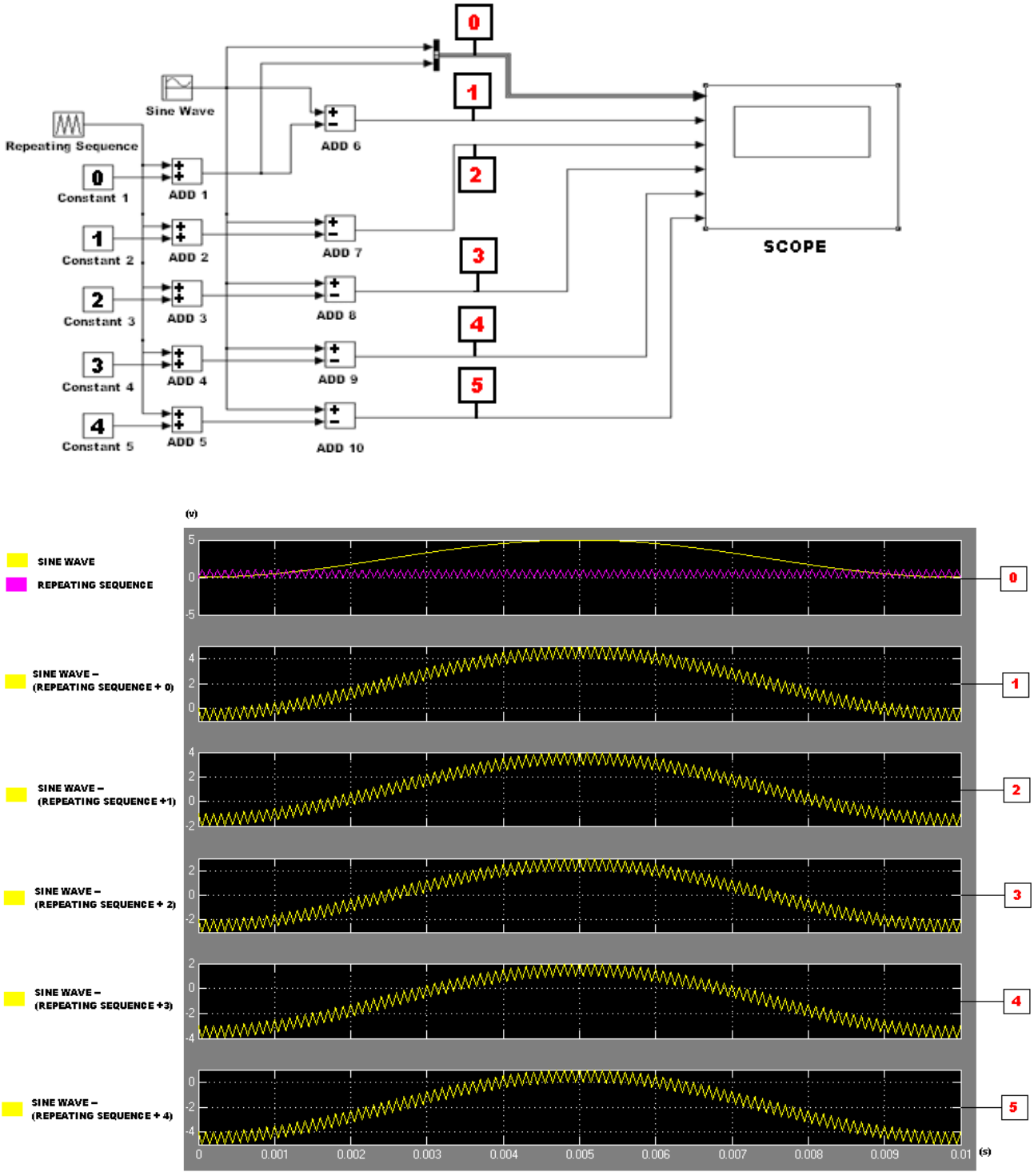


Figura 3.10: Diagrama y simulación de la comparación de las 2 señales

3.2.3) Paso por cero de cada modulo

El siguiente paso, es poner el comparador a la señal resultante de cada módulo, es decir, es el paso por cero de las señales de cada módulo. Si la señal resultante, de la comparación entre la señal moduladora y portadora, es menor que 0, la señal se pondrá a 0; si por el contrario esta señal es mayor o igual que 0, la señal se pondrá a 1.

Para conseguir esto, a continuación de la señal resultante de cada módulo pondremos el siguiente bloque:

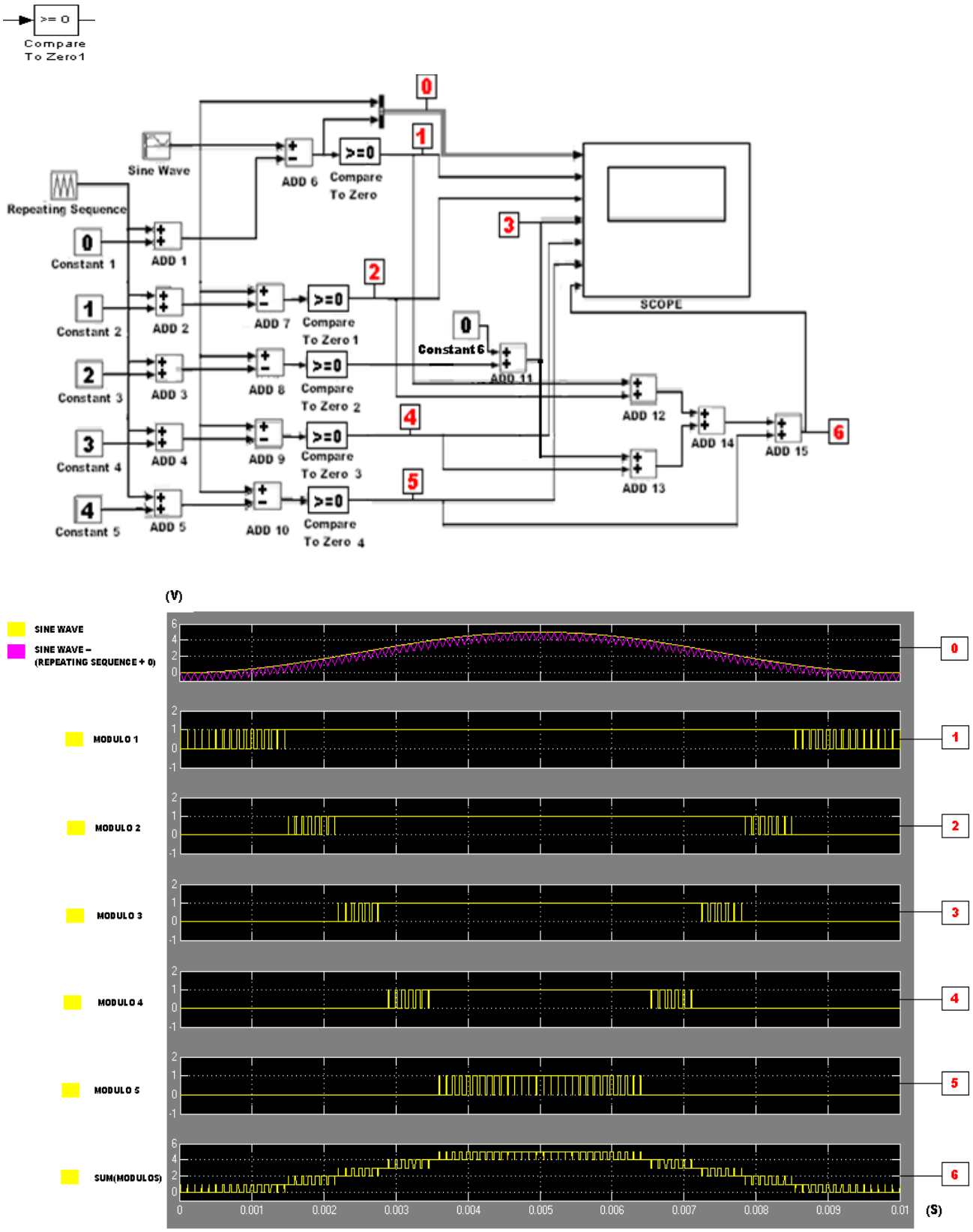


Figura 3.11: Diagrama y simulación del paso por cero de la resultante de la comparación de cada modulo

Observamos en la figura 3.11, que ya se ha obtenido el disparo de cada módulo. Vemos el sumatorio de los disparos de cada módulo, y podemos comprobar que salen correctamente. No se intercalan unos con otros, cuando termina un módulo su parpadeo empieza el del otro, y así sucesivamente.

3.2.4) Estudio del tiempo de encendido de cada módulo

Ahora surge un nuevo reto, como se puede observar en la figura 3.11: el tiempo que se mantiene encendido o apagado (al margen del parpadeo descrito) es distinto en cada módulo. Es necesario que todos los módulos estén el mismo tiempo encendidos y apagados, para que todos ellos tengan el mismo grado de utilización.

Para ello vamos a crear una fuente auxiliar, para controlar y combinar los dos bloques, de los ángulos de conmutación de cada módulo. El objetivo es que a mitad del periodo de la señal moduladora, seleccionar el ángulo de conmutación necesario para cumplir con el requisito de que cada módulo esté encendido el mismo tiempo a lo largo de un semiciclo de red.

Observamos la figura 3.11 y las comparaciones necesarias son las siguientes:

- MODULO 1 con el MODULO5.
- MODULO 2 con el MODULO4.
- MODULO 3, se quedaría como esta, es el céntrico y tendría el tiempo de apagado y encendido que deseamos.
- MODULO4 con el MODULO2.
- MODULO5 con el MODULO1.

Vamos a explicar gráficamente la comparación del MODULO 1 con el MODULO5:

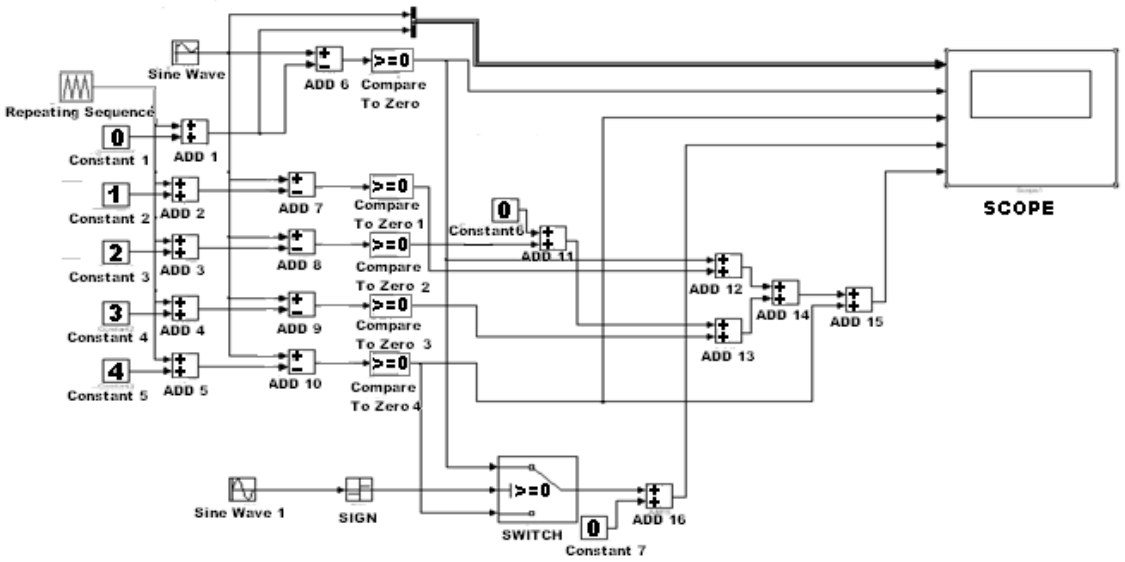


Figura 3.12: Diagrama en el que comparamos el MODULO 1 y el MODULO5

Los valores de nuestra nueva fuente son:

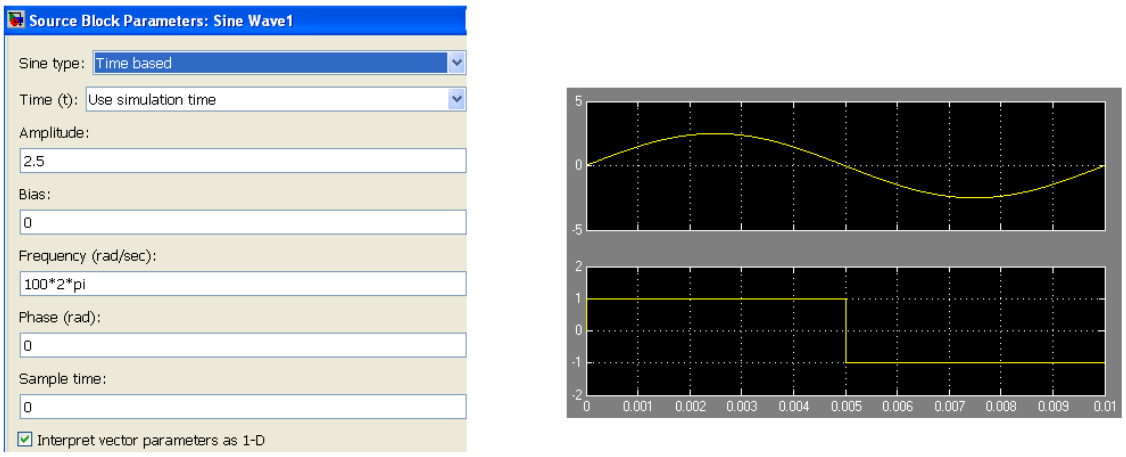


Figura 3.13: Parámetros de la fuente auxiliar para obtener el mismo tiempo de encendido y apagado de cada modulo

Vemos en la figura 3.12, que después de la "SINE WAVE 1", lo hemos pasado por el bloque SIGN, para hacerla una onda cuadrada. Mantenemos la misma amplitud y frecuencia que la señal "SINE WAVE".

La diferencia entre las 2 señales sinusoidales las encontramos en las BIAS y en la FASE:

- La Sine Wave tiene 2.5 de amplitud y 2.5 BIAS, lo que conseguimos con esto es desplazar esta señal 2.5 hacia arriba en el eje vertical, y llevarla hacia el cuadrante positivo, por lo que al desplazarla hacia arriba, el valor máximo de esta señal será de 5(v).

  A parte, la Sine Wave esta desplazada $-\pi/2$, para conseguir que empiece y acabe en 0, y que alcance su máxima amplitud en la mitad del periodo, para que sea efectiva las comparaciones con las distintas señales triangulares de cada módulo y conseguir el parpadeo de los led.

- La Sine Wave1 tiene 2.5 de amplitud y 0 BIAS, por lo que tenemos una onda sinusoidal con su parte positiva y negativa, sin desplazamientos de la señal (BIAS=0) con un rango de +/-2.5(v).

  La fase de Sine Wave1 es cero. La finalidad de esta señal es que a mitad de periodo entre en la parte negativa y así poder hacer las comparaciones entre módulos anteriormente descritas.

Vamos a explicar una de las secuencias: en el modulo1, el primer bloque de parpadeo, quedaría tal cual está. A mitad de periodo, cuando la señal Sine Wive1, entre a la parte negativa, empezar a coger datos del modulo5, y así sucesivamente con todas las comparaciones entre módulos descritas.

Vamos a ver cómo hemos conseguido en SIMULINK representar lo descrito:

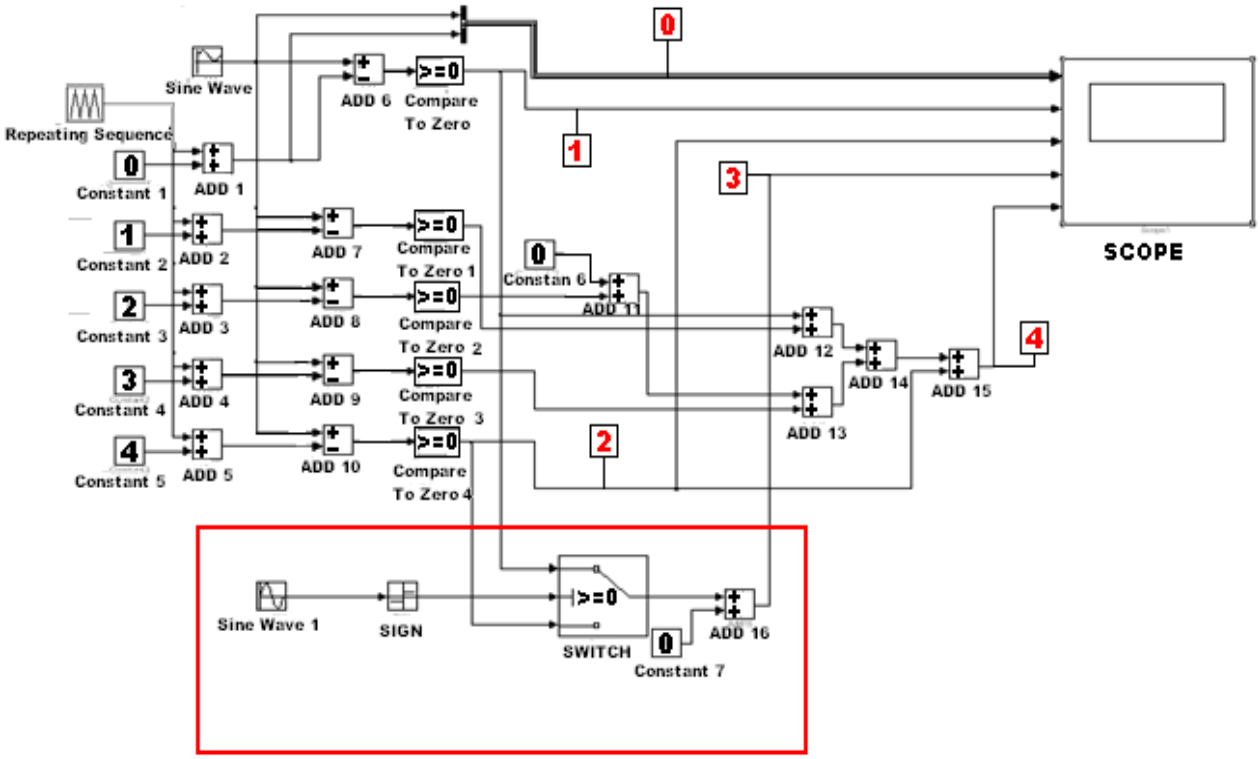
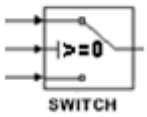


Figura 3.14 Bloque para que el módulo 1 este el mismo tiempo apagado y encendido

El bloque que hemos utilizado, para cuando Sine Wave1 este en la parte positiva, coger la mitad del periodo del módulo 1. Y, cuando Sine Wive1 este en la parte negativa, coger la mitad del periodo del módulo 5, es:



Este bloque sirve, para cuando la señal que se introduce en la patilla del medio es (>=0), obtenemos a la salida, la señal que metemos en la patilla de arriba. Si por el contrario, si la señal que introducimos en la patilla del medio, es (<0), a la salida tendremos, la señal introducida en la patilla de abajo.

Resultado grafico de la FIGURA 3.14:

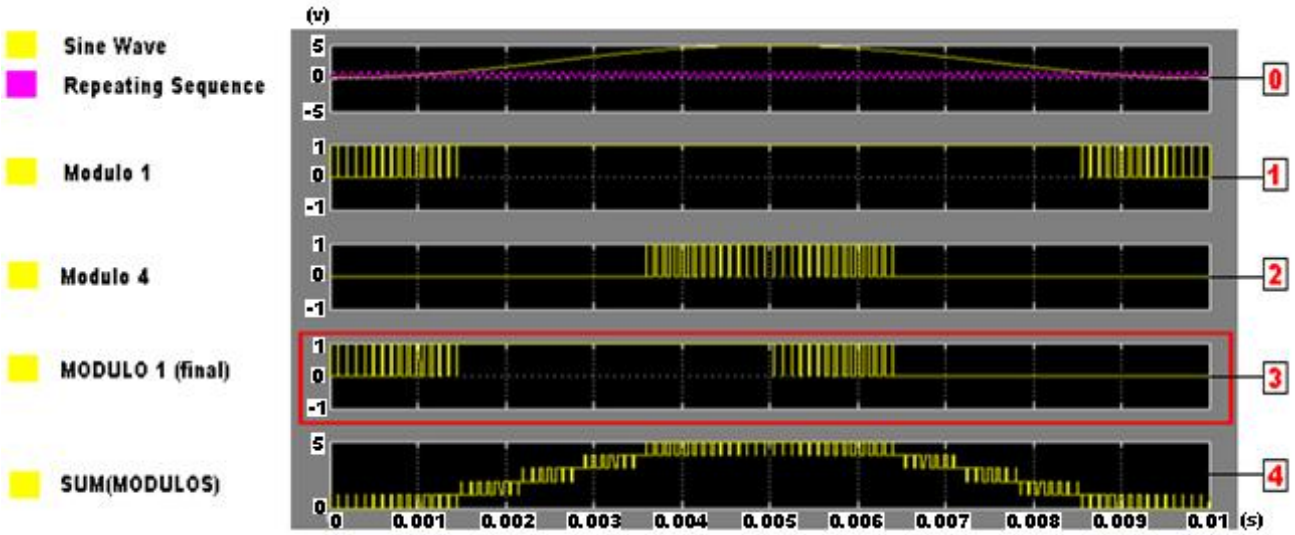


Figura 3.15: Simulación de la resultante de la comparación del MODULO 1 y el MODULO 5

3.3) RESULTADOS Y CONCLUSIONES

A continuación vamos a ver las representaciones en SIMULINK y las correspondientes graficas de todos los módulos, y podremos comprobar como en todos los módulos tenemos el mismo tiempo de encendido y apagado de los led.

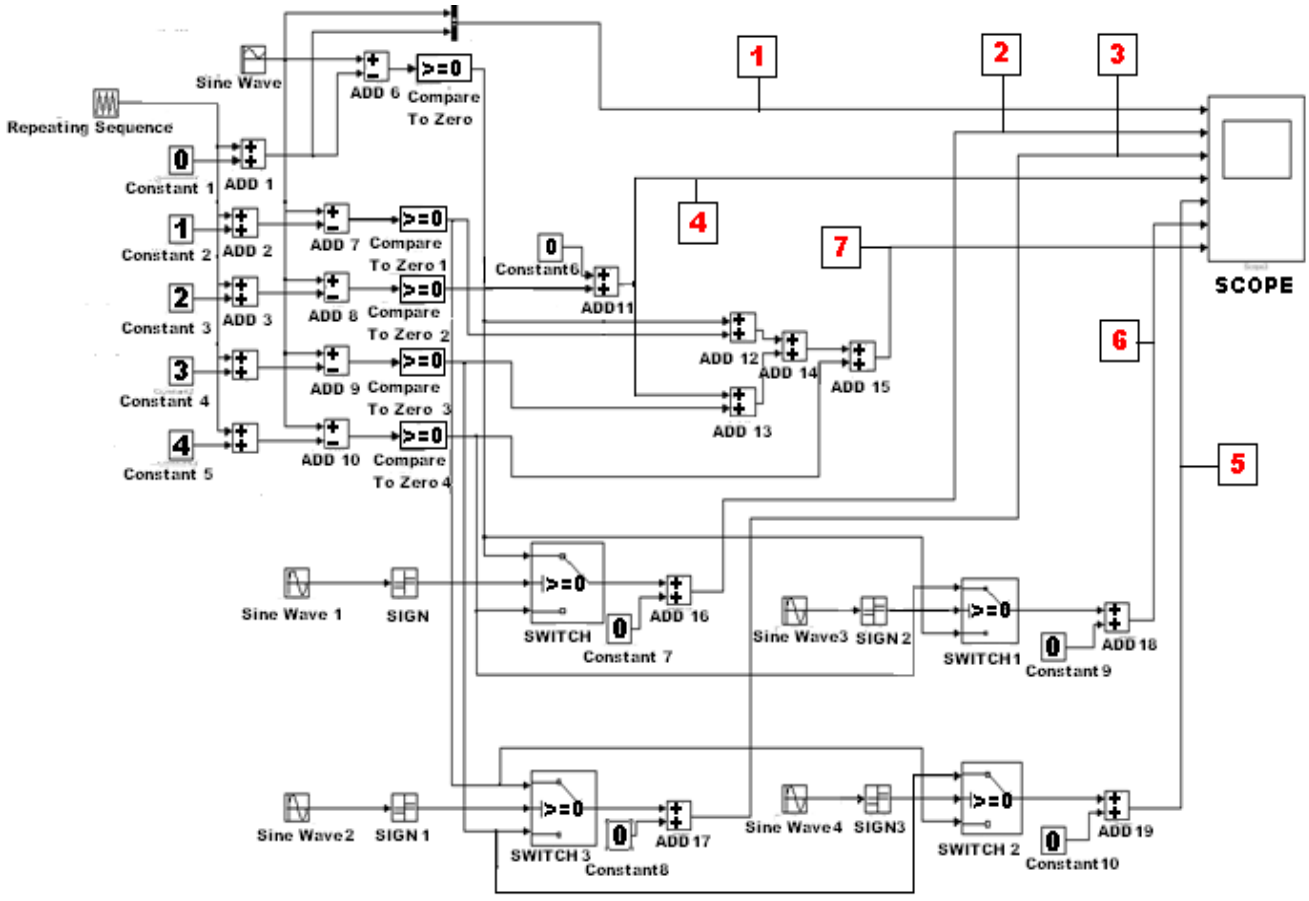


Figura 3.16 Diagrama de la comparación entre módulos para conseguir el mismo tiempo de encendido y apagado

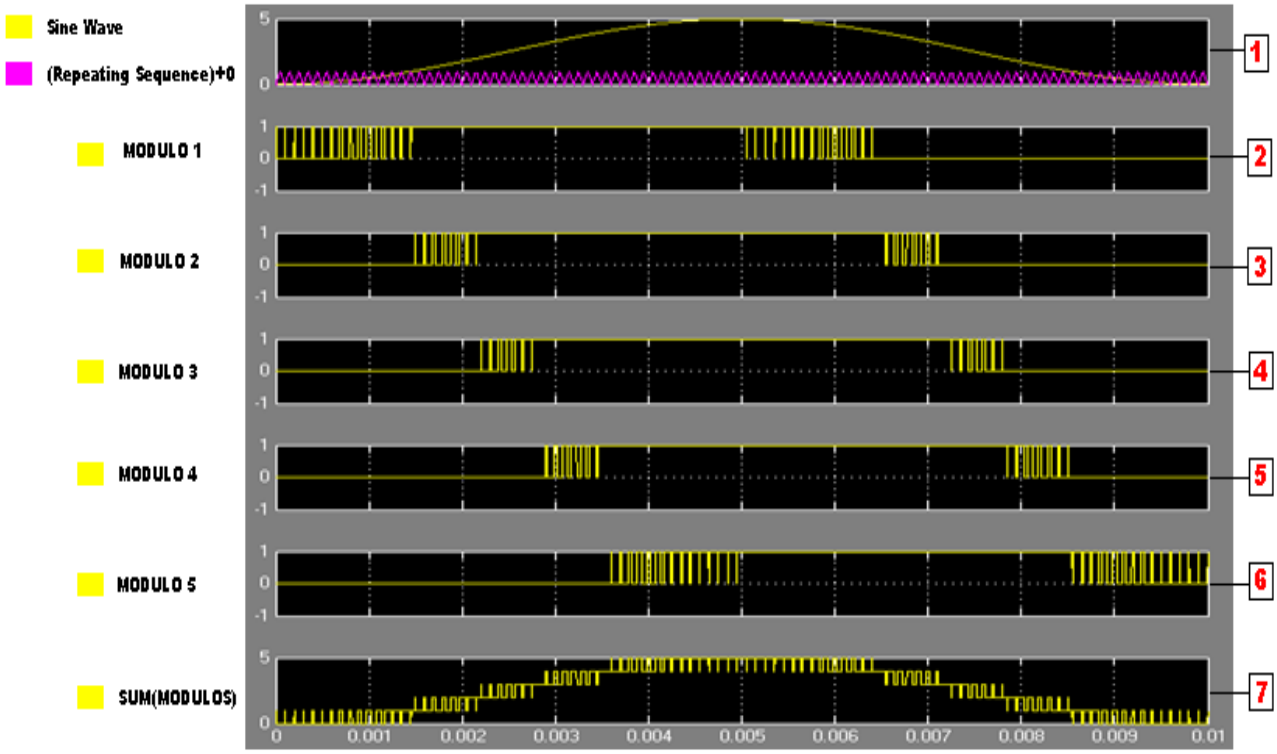


Figura 3.17: Simulación de la comparación entre módulos para conseguir el mismo tiempo de encendido y apagado

Podemos observar que hemos conseguido nuestro objetivo, el parpadeo de todos los módulos, y que todos ellos estén el mismo tiempo de encendido y apagado (fuera de lo que es el parpadeo).

Posteriormente vamos a sacar los datos del parpadeo en una tabla, es decir, en el instante de tiempo en el que cada módulo se produce el cambio de 1 a 0, y de 0 a 1.

Para ello hay que preparar un espacio de trabajo (workspace) en el diagrama que hemos realizado en SIMULINK, como podemos observar en la figura 3.18, y posteriormente en MATLAB extraeremos esos datos mediante unas líneas de código.

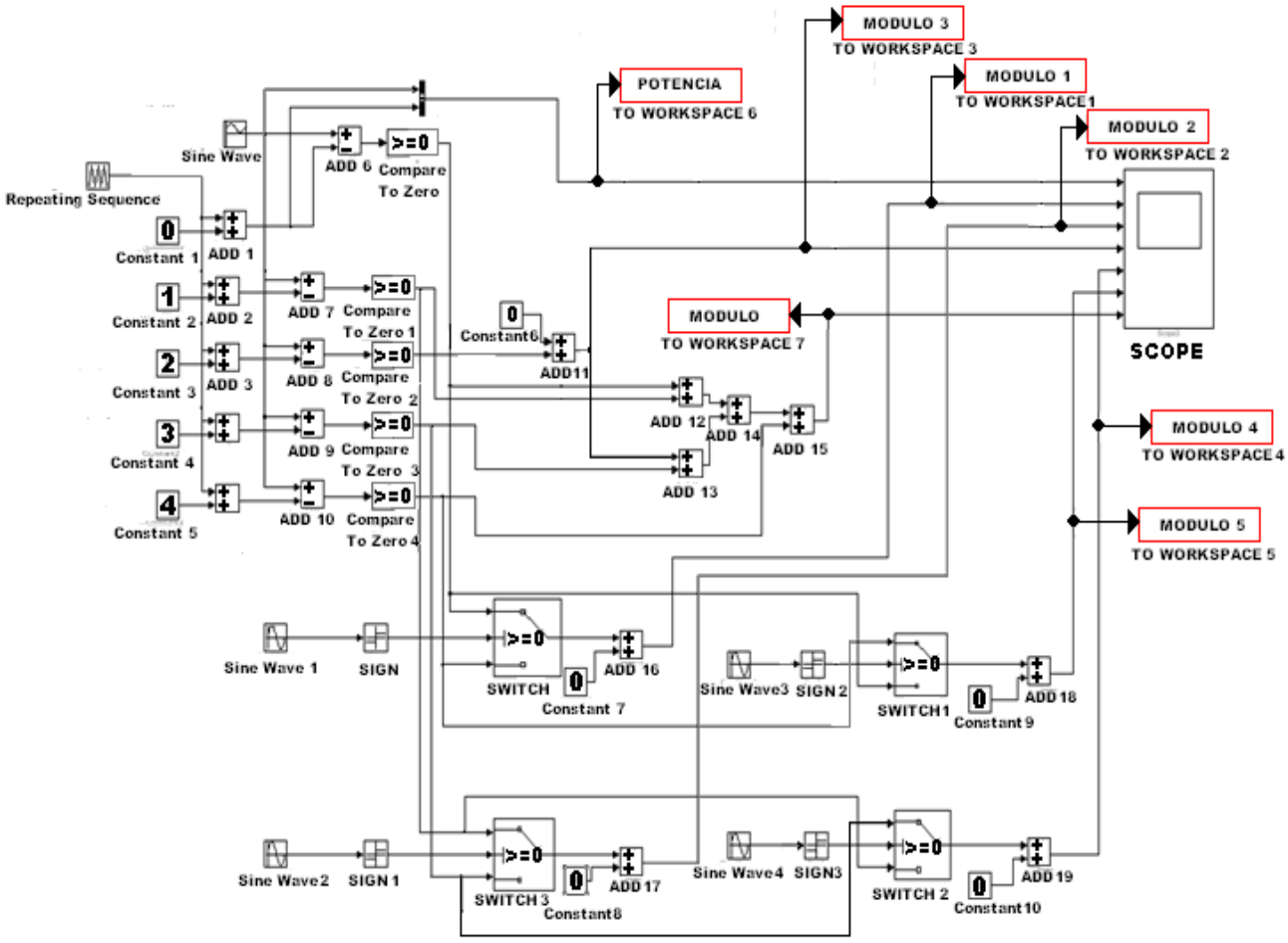


Figura 3.18: Diagrama final de la modulación PWM

Las líneas de código, que hay que escribir en MATLAB, para extraer el instante de tiempo en el que se produce un flanco, y que tipo de flanco se trata, son los siguientes:

Ej: para el módulo 1 :

```
diffmodulo=diff(modulo1);

vector=find(diffmodulo)
```

- `diffmodulo=diff(modulo1);` A “diffmodulo” le asignamos un vector en donde nos marcara con -1 y 1 donde se producirán los cambios de modulo1,y con un 0 donde no se produce ningún cambio, es decir, “diff”, va calculando la diferencia entre todos los valores(x2-x1, x3-x2….). Si el cambio de valor ha sido de 1 a 0(flanco de bajada),el vector “diffmodulo” nos marcara con -1(ya que sería:0-1=-1) ese cambio que se ha producido en ese instante de tiempo. “Diffmodulo”, también nos marcara con 1(ya que seria 1-0=1), cuando modulo1 pase de 0 a 1(flanco de subida).Y en las demás posiciones del vector 0, (ya que la diferencia seria 0-0=0).

- `vector=find(diffmodulo)`. “Find” localiza todos los elementos que no sean 0, en el vector diffmodulo, y estos elementos son mostrados en el vector. Así que “vector”, nos muestra los instantes de tiempo, donde los elementos de “diffmodulo” no sean 0.

Vamos a ver la tabla con los datos correspondientes a cada módulo:

modulo1		modulo2		modulo3		modulo4		modulo5	
1	0	1499	1	2199	1	2895	1	3596	1
100	1	1502	0	2202	0	2907	0	3605	0
101	0	1593	1	2292	1	2988	1	3691	1
200	1	1609	0	2311	0	3015	0	3712	0
201	0	1687	1	2385	1	3081	1	3785	1
298	1	1716	0	2419	0	3123	0	3818	0
303	0	1780	1	2477	1	3174	1	3880	1
397	1	1824	0	2528	0	3231	0	3923	0
405	0	1873	1	2570	1	3268	1	3976	1
495	1	1932	0	2636	0	3338	0	4028	0
507	0	1967	1	2663	1	3361	1	4071	1
592	1	2040	0	2745	0	3445	0	4132	0
610	0	2060	1	2756	1	3455	1	4167	1
689	1	2148	0	7245	0	7848	0	4236	0
713	0	2153	1	7256	1	7853	1	4264	1
786	1	6546	0	7338	0	7941	0	4340	0
817	0	6556	1	7365	1	7961	1	4360	1
882	1	6640	0	7431	0	8034	0	4443	0
921	0	6663	1	7473	1	8069	1	4458	1
978	1	6733	0	7524	0	8128	0	4545	0
1026	0	6770	1	7582	1	8177	1	4555	1
1073	1	6827	0	7616	0	8221	0	4647	0
1131	0	6878	1	7690	1	8285	1	4653	1
1168	1	6920	0	7709	0	8314	0	4749	0
1236	0	6986	1	7799	1	8392	1	4752	1
1263	1	7013	0	7802	0	8408	0	4850	0
1342	0	7094	1			8499	1	4851	1
1358	1	7106	0			8502	0	4950	0
1449	0							4951	1
1452	1							8549	0
5050	0							8552	1
5051	1							8643	0
5150	0							8659	1
5151	1							8738	0
5249	0							8765	1
5252	1							8833	0
5348	0							8870	1
5354	1							8928	0
5446	0							8975	1
5456	1							9023	0
5543	0							9080	1
5558	1							9119	0
5641	0							9184	1
5661	1							9215	0
5737	0							9288	1
5765	1							9312	0
5834	0							9391	1
5869	1							9409	0
5930	0							9494	1
5973	1							9506	0
6025	0							9596	1
6078	1							9604	0
6121	0							9698	1
6183	1							9703	0
6216	0							9800	1
6289	1							9801	0
6310	0							9900	1
6396	1							9901	0
6405	0							10000	1

Figura 3.19: Tabla con el instante de tiempo y el valor de cada modulo

NOTA: - "0"→ flanco de bajada. Cuando los datos pasan de 1 a 0(se apaga el led).
- "1" → flanco de subida. Cuando los datos pasan de 0 a 1(se enciende el led).

Una vez que ya tenemos los datos de cada módulo, vamos a meterlos todos en un vector, en donde se indicara el instante de tiempo, en donde se produzca un flanco en cada uno de los 5 módulos, y también se indicara el estado de los led de cada módulo.

Ejemplo: En el módulo 1, en el instante 100, el led está encendido, ya que está a 1, entonces en nuestro vector general de todos los módulos, quedaría de la siguiente forma, nos indicaría que en 100 habría el siguiente dato:"00001".

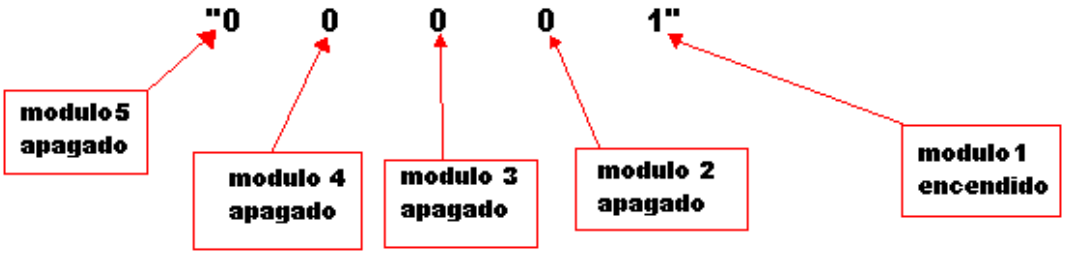


Figura 3.20: Posición de cada módulo en el vector

El vector que engloba a todos los módulos, lo hemos llamado "modulo", y quedaría de la siguiente forma:

MODULO:							
1	.00000	2385	.00111	4951	.11111	7582	.11100
100	.00001	2419	.00011	5050	.11110	7616	.11000
101	.00000	2477	.00111	5051	.11111	7690	.11100
200	.00001	2528	.00011	5150	.11110	7709	.11000
201	.00000	2570	.00111	5151	.11111	7799	.11100
298	.00001	2636	.00011	5249	.11110	7802	.11000
303	.00000	2663	.00111	5252	.11111	7848	.10000
397	.00001	2745	.00011	5348	.11110	7853	.11000
405	.00000	2756	.00111	5354	.11111	7941	.10000
495	.00001	2895	.01111	5446	.11110	7961	.11000
507	.00000	2907	.00111	5456	.11111	8034	.10000
592	.00001	2988	.01111	5543	.11110	8069	.11000
610	.00000	3015	.00111	5558	.11111	8128	.10000
689	.00001	3081	.01111	5641	.11110	8177	.11000
713	.00000	3123	.00111	5661	.11111	8221	.10000
786	.00001	3174	.01111	5737	.11110	8285	.11000
817	.00000	3231	.00111	5765	.11111	8314	.10000
882	.00001	3268	.01111	5834	.11110	8392	.11000
921	.00000	3338	.00111	5869	.11111	8408	.10000
978	.00001	3361	.01111	5930	.11110	8499	.11000
1026	.00000	3445	.00111	5973	.11111	8502	.10000
1073	.00001	3455	.01111	6025	.11110	8549	.00000
1131	.00000	3596	.11111	6078	.11111	8552	.10000
1168	.00001	3605	.01111	6121	.11110	8643	.00000
1236	.00000	3691	.11111	6183	.11111	8659	.10000
1263	.00001	3712	.01111	6216	.11110	8738	.00000
1342	.00000	3785	.11111	6289	.11111	8765	.10000
1358	.00001	3818	.01111	6310	.11110	8833	.00000
1449	.00000	3880	.11111	6396	.11111	8870	.10000
1452	.00001	3923	.01111	6405	.11110	8928	.00000
1499	.00011	3976	.11111	6546	.11100	8975	.10000
1502	.00001	4028	.01111	6556	.11110	9023	.00000
1593	.00011	4071	.11111	6640	.11100	9080	.10000
1609	.00001	4132	.01111	6663	.11110	9119	.00000
1687	.00011	4167	.11111	6733	.11100	9184	.10000
1716	.00001	4236	.01111	6770	.11110	9215	.00000
1780	.00011	4264	.11111	6827	.11100	9288	.10000
1824	.00001	4340	.01111	6878	.11110	9312	.00000
1873	.00011	4360	.11111	6920	.11100	9391	.10000
1932	.00001	4443	.01111	6986	.11110	9409	.00000
1967	.00011	4458	.11111	7013	.11100	9494	.10000
2040	.00001	4545	.01111	7094	.11110	9506	.00000
2060	.00011	4555	.11111	7106	.11100	9596	.10000
2148	.00001	4647	.01111	7245	.11000	9604	.00000
2153	.00011	4653	.11111	7256	.11100	9698	.10000
2199	.00111	4749	.01111	7338	.11000	9703	.00000
2202	.00011	4752	.11111	7365	.11100	9800	.10000
2292	.00111	4850	.01111	7431	.11000	9801	.00000
2311	.00011	4851	.11111	7473	.11100	9900	.10000
2385	.00111	4950	.01111	7524	.11000	9901	.00000
						10000	.10000

Figura 3.21: Tabla de los vectores de cada modulo

Conclusiones:

El periodo de los instantes de conmutación de los 5 módulos es de 10(ms), ya que tiene que coincidir con el periodo de la señal de salida de nuestro circuito, como se verá en el siguiente capítulo. Todos los módulos tienen que estar los mismos tiempos encendidos y apagados, para que tengan el mismo grado de utilización.

En vista a futuros proyectos: En este proyecto hemos realizado una PWM con 5 niveles, ya que el tubo de led, disponíamos de un determinado número de led, se tomó la decisión de dividirlo en 5 módulos. Pero en vista a proyectos futuros, se puede ampliar a varios niveles más, con varios módulos.

Otro idea para proyectos futuros seria incrementar el número de instantes de conmutación, variando la frecuencia de la señal portadora, o de ampliar el tiempo en el que el modulo permanece a uno a cero, variando la amplitud de la señal sinusoidal.

4 CIRCUITO DE SINCRONIZACION CON LA RED.

4.1 OBJETIVO

El objetivo de este circuito, es conseguir a la salida de nuestro circuito una señal cuadrada de 3.3 (v), que este sincronizada con nuestra señal de red, es decir, crear una señal, que controle el paso por cero de la señal de red. Así, cada vez que empiece la señal de salida que vamos a crear, se van a ir introduciendo los datos de cada módulo que hemos creado anteriormente, mediante un código de sincronismo realizado en la FPGA.Y que cuando acabe el periodo de la señal de salida de nuestro circuito, coincida con el fin del periodo de los instantes de conmutación y así conseguiremos, que nuestros módulos se vayan encendiendo y apagando de forma sincronizada con la señal de red.

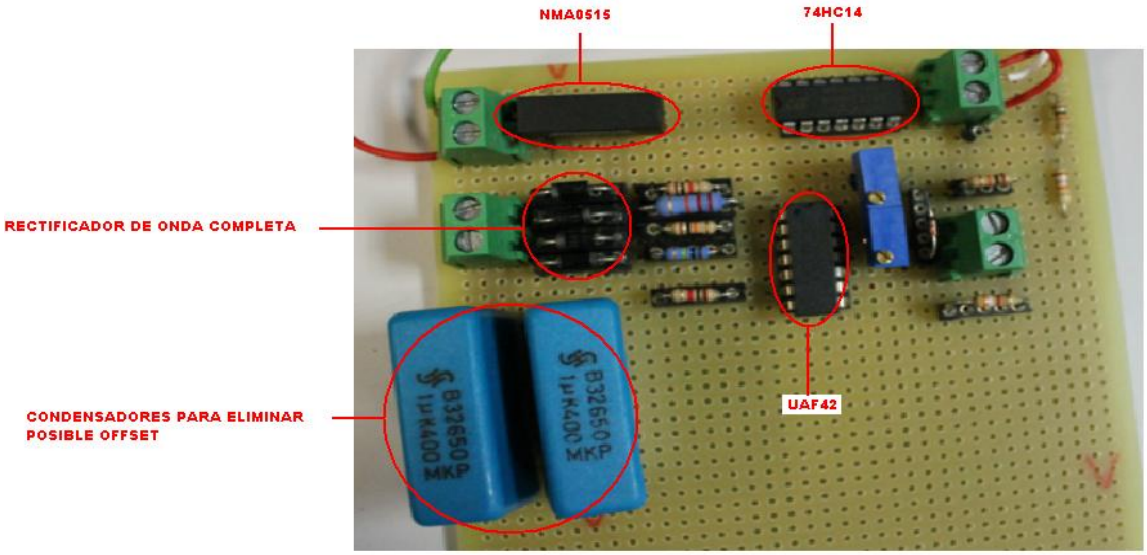
La amplitud de la señal de salida tiene que ser de 3.3 (v), para poder introducirla en la FPGA, y poder sincronizarla mediante un código con los instantes de conmutación que anteriormente se han descrito.

Vamos a ir explicando cada parte del circuito y los resultados que vayan saliendo hasta conseguir nuestro objetivo.

4.2 ESQUEMATICO DEL CIRCUITO

Análisis de cada parte del circuito, que posteriormente analizaremos su correspondiente función.

-Esquemático general del circuito:



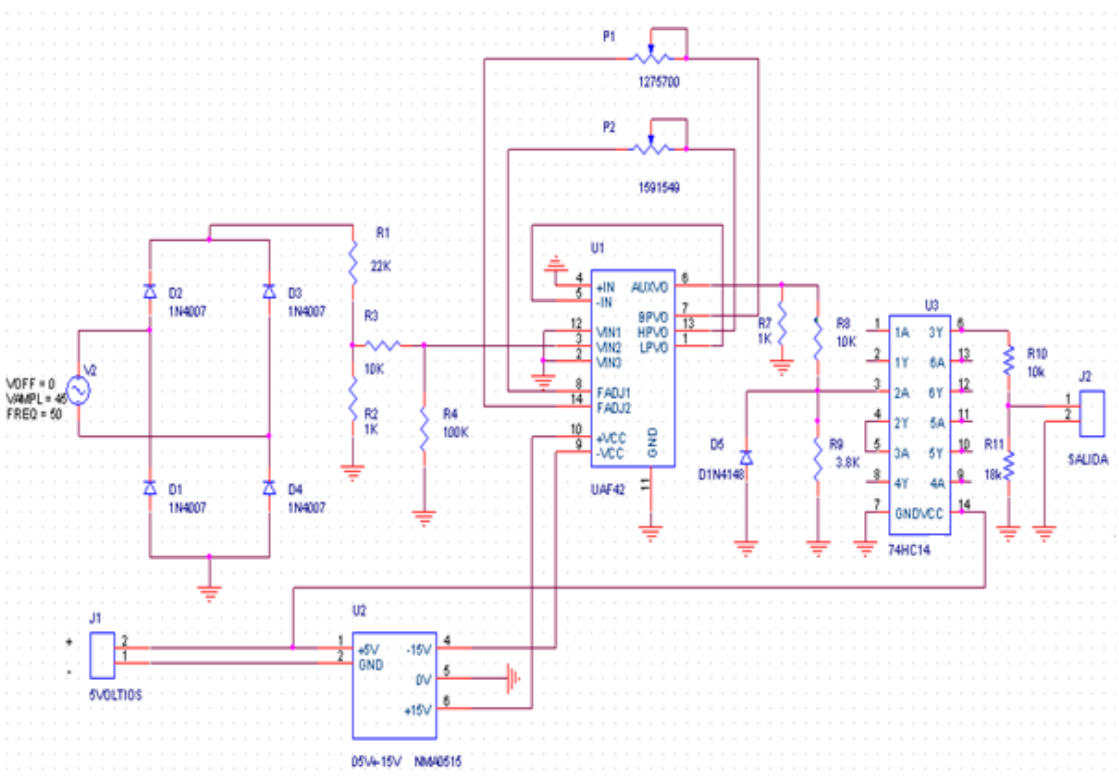


Figura 4.1: Circuito completo

4.2.1) Fuente de alimentación continua y alterna

4.2.1.1) Señal de entrada continúa

Se alimentara el circuito integrado con una tensión continua de 5(v), que aparte de alimentar el circuito, servirá para mandar la tensión de 5(v) al convertidor de tensión.

Posteriormente se utiliza una fuente de alimentación continua variable de 5(v), para alimentar la FPGA . Y, de un pin de la FPGA , se saca la tensión continua que alimenta la placa.

4.2.1.2) Señal de entrada de alterna

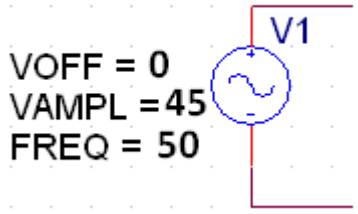


Figura 4.2: Señal de entrada

La tensión de alterna que se utiliza para realizar las pruebas pertinentes es de 45(v) de amplitud, antes de llegar a probar con una fuente de alterna más grande, la cual, necesitamos para llegar a encender los led.
Como podemos observar, la frecuencia a la entrada de nuestra señal es de 50(Hz), por lo que el periodo de nuestra señal de entrada es de 20(ms)→T=1/f.

Necesitamos que el periodo de nuestra señal de salida sea de 10(ms), como ya hemos podido comprobar, en nuestro periodo de tiempo de los disparos de led de los módulos, realizados en MATLAB.

Se requiere, que el periodo de la señal de salida sea la mitad que el de la señal de red. El objetivo es poder detectar cada paso por cero de la señal sinusoidal de red y así poder sincronizar con la red el encendido y apagado de los módulos. Para conseguir ese incremento de la frecuencia en la señal de salida, y que el periodo sea de 10(ms), se utiliza el filtro universal UAF42, con elevada Q, para generar 3 señales, con una frecuencia de 100(Hz) a partir de la señal de red rectificada. El elevado factor de calidad hace que el filtro sea más selectivo y se pueda detectar mejor la componente de 100 Hz de la señal de red rectificada.

NOTA: la tensión de OFFSET a 0.

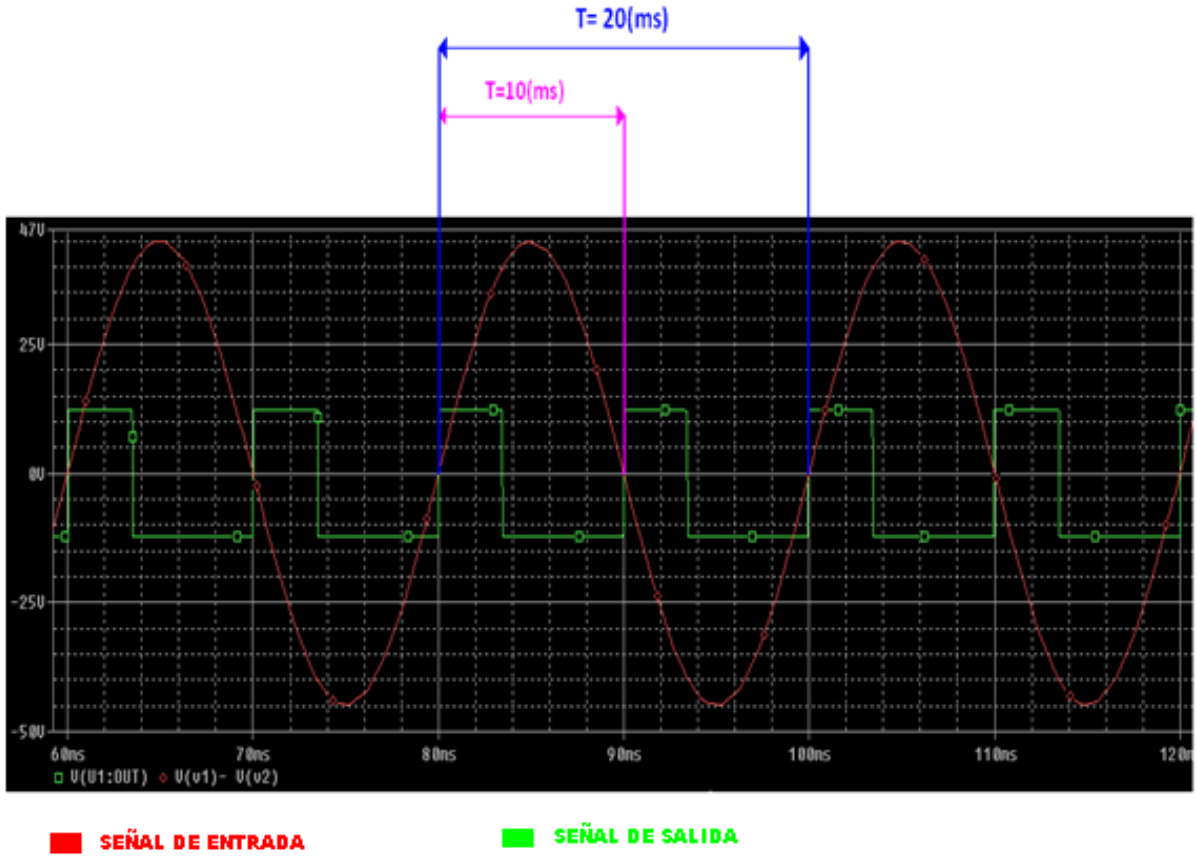


Figura 4.2: Señal de entrada y señal de salida. Frecuencia elegida

4.2.2) Rectificador

Un rectificador es un circuito que convierte una señal de CA en una señal unidireccional. Los diodos son utilizados de manera constante en rectificadores.

Los diodos rectificadores:

Deben ser diodos con unas características especiales. De hecho, existe un subgrupo de diodos llamados así, rectificadores. Los diodos rectificadores deben poder ser capaces de soportar de

forma continua valores de corriente que, según de qué aplicaciones se trate, puede llegar a ser elevada o muy elevada. Además, deben soportar picos de corriente varias veces mayores que su corriente nominal máxima de funcionamiento. En cuanto a las características de tensión, es normal que puedan trabajar con tensiones inversas de algunas centenas de voltios. Tomemos como ejemplo un diodo rectificador muy difundido, el **1N4007**, el cual hemos utilizado para realizar nuestro rectificador en nuestro circuito. Tiene aplicación en fuentes de alimentación de pequeña potencia de salida.

Sus principales características son:

- Picos repetitivos de tensión inversa: 1000V máximo.
- Picos no repetitivos de tensión inversa: 1200V máximo.
- Tensión inversa máxima de forma continua: 700V.
- Corriente nominal directa máxima: 1A.
- Picos de corriente directa no repetitivos: 30A máximo.

Los rectificadores pueden ser de media onda y de onda completa: la gran diferencia es que el rectificador de media onda solo deja pasar a la salida la parte positiva de la señal de entrada, la parte negativa la deja a 0; mientras que el rectificador de onda completa deja pasar la onda positiva de entrada y la parte negativa, la traslada al semiciclo positivo.

Lo vamos a representar gráficamente:

-RECTIFICADOR DE MEDIA ONDA:

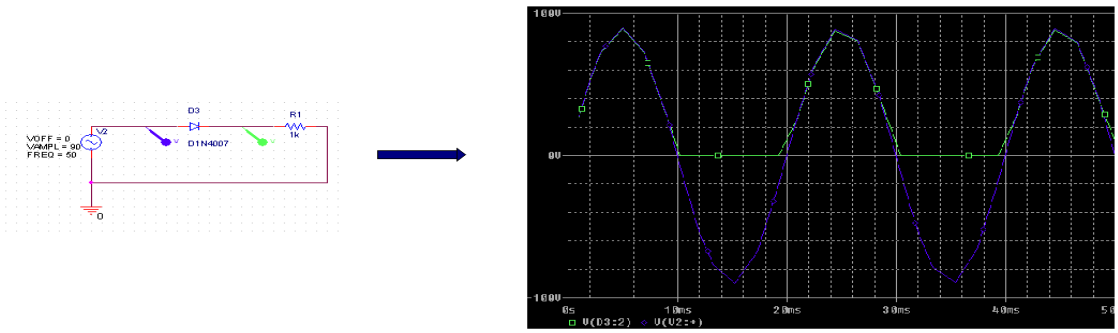


Figura 4.3: Representación teórica de un rectificador de media onda

-RECTIFICADOR DE ONDA COMPLETA:

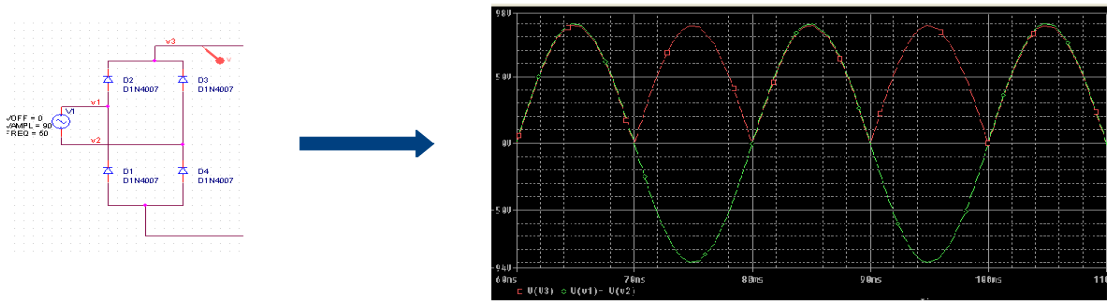


Figura 4.4: Representación teórica de un rectificador de onda completa

En el caso de nuestro proyecto, hemos optado por el rectificador de onda completa, ya que nuestro objetivo es controlar el paso por cero de nuestra señal, y con este rectificador nos va a resaltar cada vez que pase por cero nuestra señal de entrada.

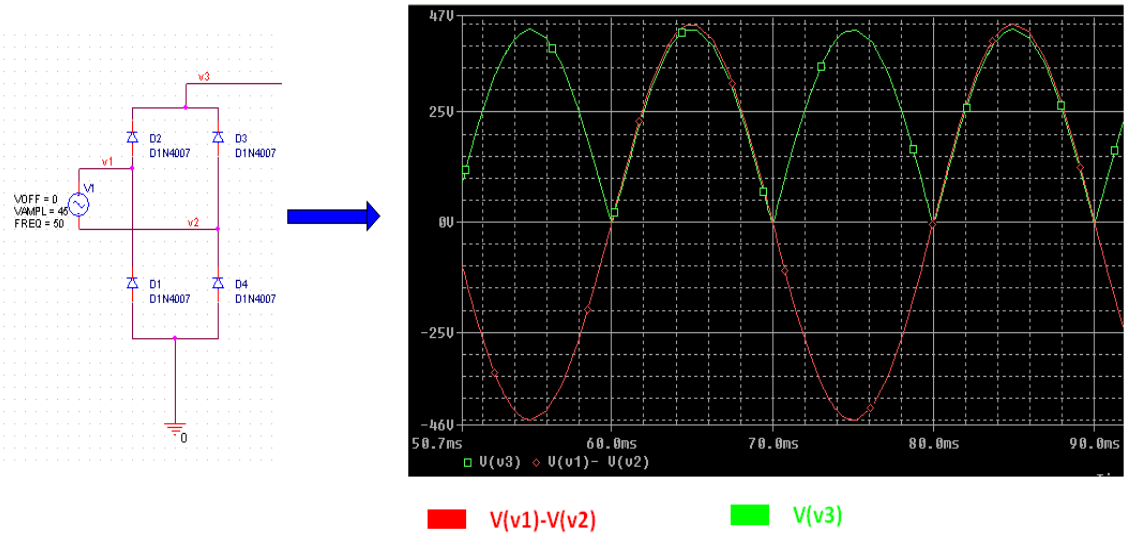


Figura 4.5: Rectificador de onda completa

Circuito de onda completa tipo puente (puente de Graetz)

El circuito monofásico de onda completa tipo puente es el más utilizado. Se aplica generalmente, cuando interesa que la tensión de salida, sea de valor similar al de la tensión eficaz de alimentación.

Para determinar los valores de los componentes utilizados, es importante conocer el tipo de carga, a que será sometido el rectificador (resistiva, capacitiva o inductiva). La figura 4.5, presenta un circuito, rectificador monofásico de onda completa tipo puente, en el cual el ángulo de conducción de los diodos vale 180˚.

En este circuito puente, se emplean cuatro diodos: cuando la tensión de entrada V1 es positiva, conducen los diodos D2 y D4; cuando es negativa, conducen D1 y D3, y los otros dos están en corte.

SEMICICLO POSITIVO

Conducen D2 y D4, mientras que D1 y D3 están en corte.

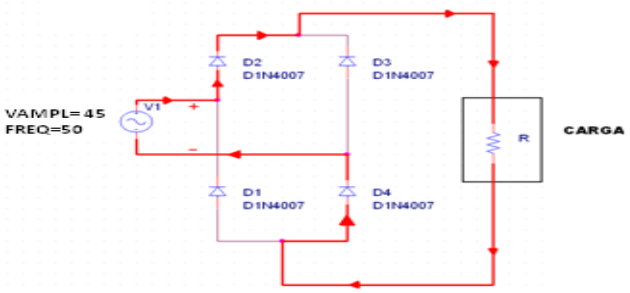


Figura 4.6: Semiciclo positivo del puente de diodos

SEMICICLO NEGATIVO

Conducen D1 y D3, mientras que D2 y D4 están en corte.

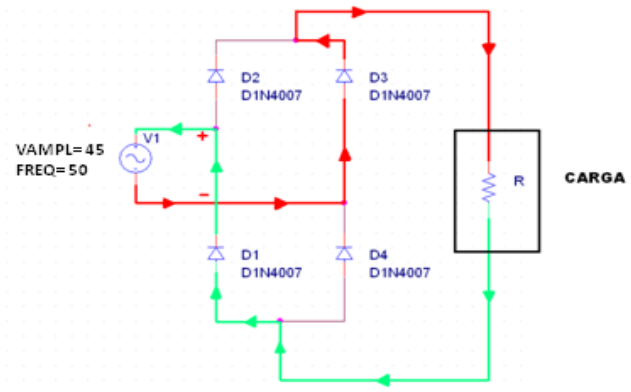
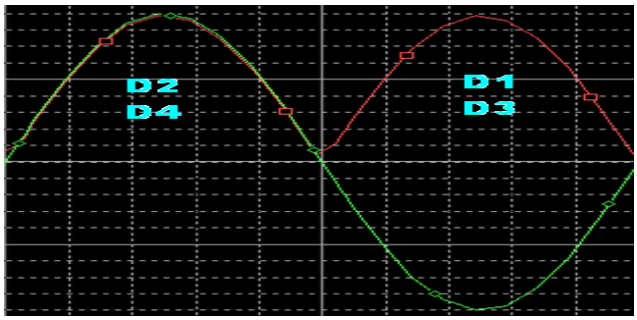


Figura 4.7: Semiciclo negativo del puente de diodos

La representación gráfica, de conducción de cada diodo, en nuestra placa, quedaría de la siguiente forma:



4.2.3) Acoplamiento del UAF42. Alimentación del UAF42. UAF42

Acoplamiento UAF42

Después del rectificador de onda completa, se pasa la señal por un divisor de tensión, para reducir la señal rectificada, ya que esta es muy elevada, y así poder adaptarla a la entrada del UAF42.

La finalidad de pasar la señal rectificada por el UAF42, es, que este filtro nos va a generar 3 señales de salida de 100(Hz), de las cuales se elige aquella señal que coincida con el pasa por cero, de la señal de entrada sinusoidal, que alimenta a nuestro circuito. Se elige una de estas señales, gracias al ajuste de los potenciómetros que observamos en la figura, que hará que coincida en el paso por cero con la señal de entrada

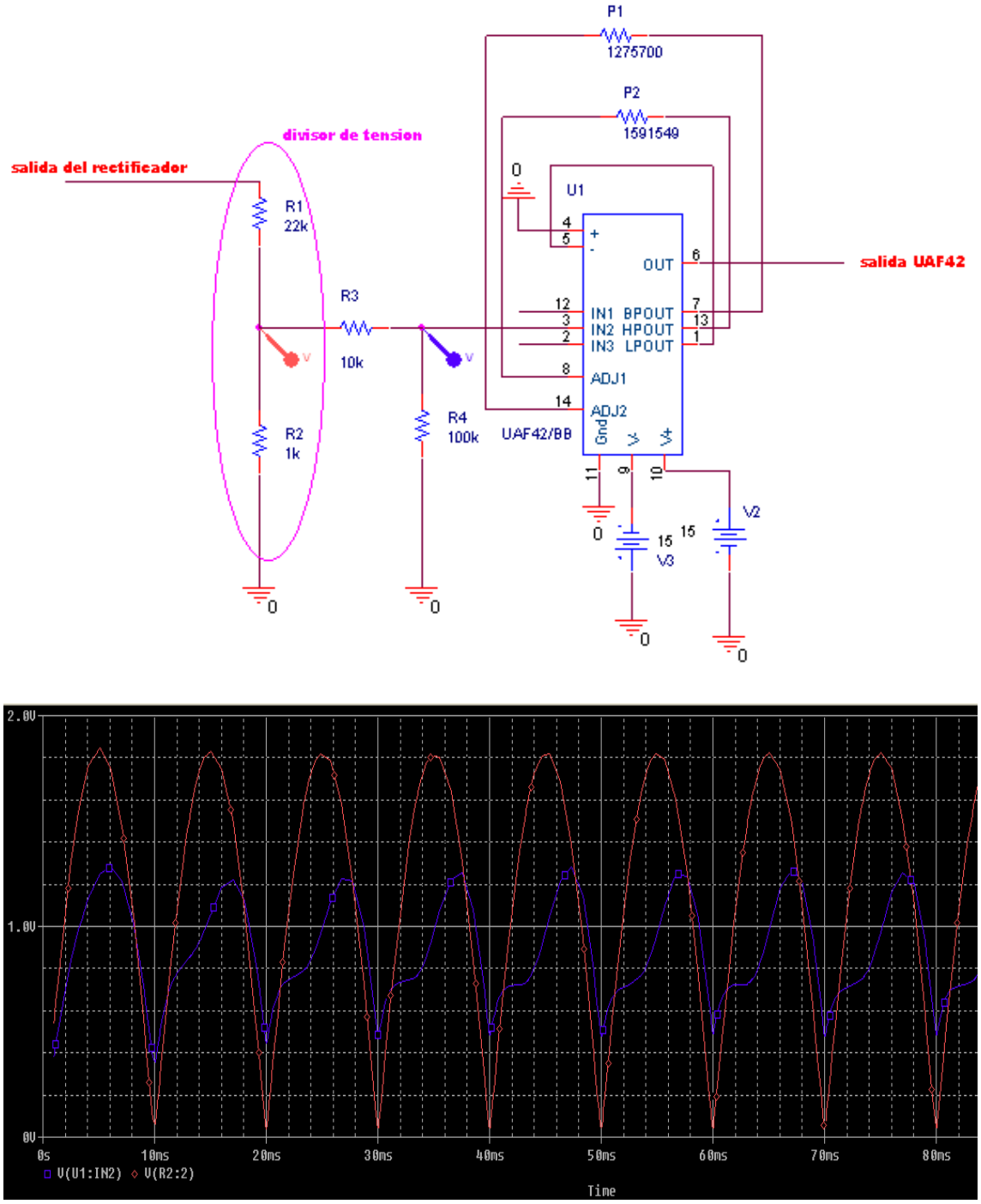


Figura 4.8: Señal de entrada del UAF42

A continuación, se explica las características y los resultados convenientes del UAF42:

ALIMENTACION DEL UAF42

Este integrado se alimenta a +/- 15(V) de continua. Para conseguir este voltaje se utiliza un convertidor de tensión, que convierta los 5(v) de continua con el que se alimenta la placa y los convierta en +/-15(v).

NMA0515SC: Alimentado a 5(v) de continua.

UAF42: Alimentado a +/-15(v) de continua.

Como se observa en la figura 4.19, el convertidor elegido es el NMA0515SC, ya que nuestra fuente de continua es de 5(v) y se necesita una tensión de 15(v) para alimentar el UAF42.

NMA 5V, 12V & 15V Series

Isolated 1W Dual Output DC/DC Converters

SELECTION GUIDE

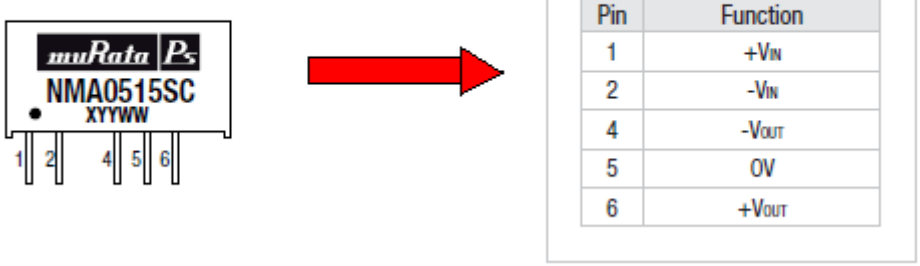
Order Code	Nominal Input Voltage	Output Voltage	Output Current	Input Current at Rated Load	Efficiency	Isolation Capacitance	MTTF[1]	Package Style
	V	V	mA	mA	%	pF	kHrs	
NMA0505DC	5	±5	±100	289	69	28	3103	DIP
NMA0509DC	5	±9	±55	267	75	32	2257	
NMA0512DC	5	±12	±42	260	77	34	1579	
NMA0515DC	5	±15	±33	256	78	36	1065	
NMA0505SC	5	±5	±100	289	69	28	3103	SIP
NMA0509SC	5	±9	±55	267	75	32	2257	
NMA0512SC	5	±12	±42	260	77	34	1579	
NMA0515SC	5	±15	±33	256	78	36	1065	
NMA1205DC	12	±5	±100	120	69	33	2193	DIP
NMA1209DC	12	±9	±55	113	74	46	1734	
NMA1212DC	12	±12	±42	111	75	55	1303	
NMA1215DC	12	±15	±33	110	76	54	932	
NMA1205SC	12	±5	±100	120	69	33	2193	SIP
NMA1209SC	12	±9	±55	113	74	46	1734	
NMA1212SC	12	±12	±42	111	75	55	1303	
NMA1215SC	12	±15	±33	110	76	54	932	
NMA1505DC	15	±5	±100	91	71	39	1941	DIP
NMA1512DC	15	±12	±42	87	78	68	790	
NMA1515DC	15	±15	±33	84	80	84	523	
NMA1505SC	15	±5	±100	91	71	39	1941	SIP
NMA1512SC	15	±12	±42	87	78	68	790	
NMA1515SC	15	±15	±33	84	80	84	523	

INPUT CHARACTERISTICS

Parameter	Conditions	Min.	Typ.	Max.	Units
Voltage range	Continuous operation, 5V input types	4.5	5	5.5	V
	Continuous operation, 12V input types	10.8	12	13.2	
	Continuous operation, 15V input types	13.5	15	16.5	
Reflected ripple current			20	40	mA p-p

Figura 4.9: Características del convertidor NMA0515SC

Las 5 patillas del convertidor, corresponde a las siguientes tensiones:



PIN CONNECTIONS - 7 PIN SIP

Pin	Function
1	+VIN
2	-VIN
4	-VOUT
5	0V
6	+VOUT

Figura 4.10: Pines del convertidor

- Vin → +/- 5(v)
- Vout→ +/-15(v)

UAF42

El UAF42 es un circuito integrado monolítico, que contiene tres amplificadores operacionales, resistores de 50 kΩ con exactitud del 0,5% y condensadores de 1.000pF de muy bajas perdidas y alta precisión ajustados por láser a 0,5%, además de un cuarto amplificador operacional auxiliar, todo ello destinado a facilitar el diseño de filtros de audio con un factor de calidad muy alto.

El circuito integrado de este filtro es el de la figura 4.12.Se observa que se generan 3 señales de salida (HIGH-PASS, BAND PASS Y LOW PASS).De estas 3 señales elegiremos la que nos convenga, es decir, aquella señal que coincida con el paso por cero de la señal sinusoidal de entrada. Al final, se observa un comparador que se utiliza para hacer, de la señal que se elija, una señal cuadrada.

CIRCUITO INTEGRADO DEL UAF42:

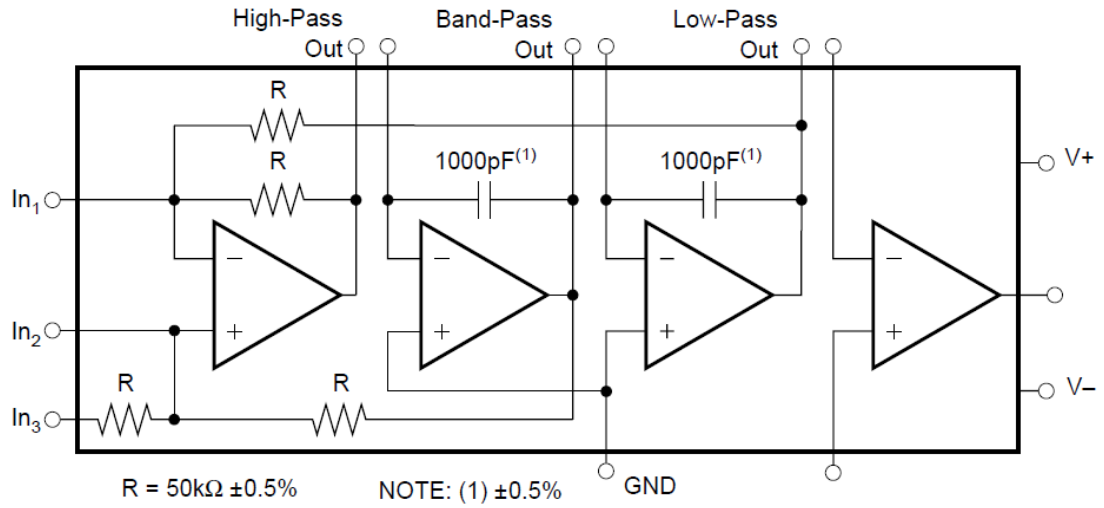
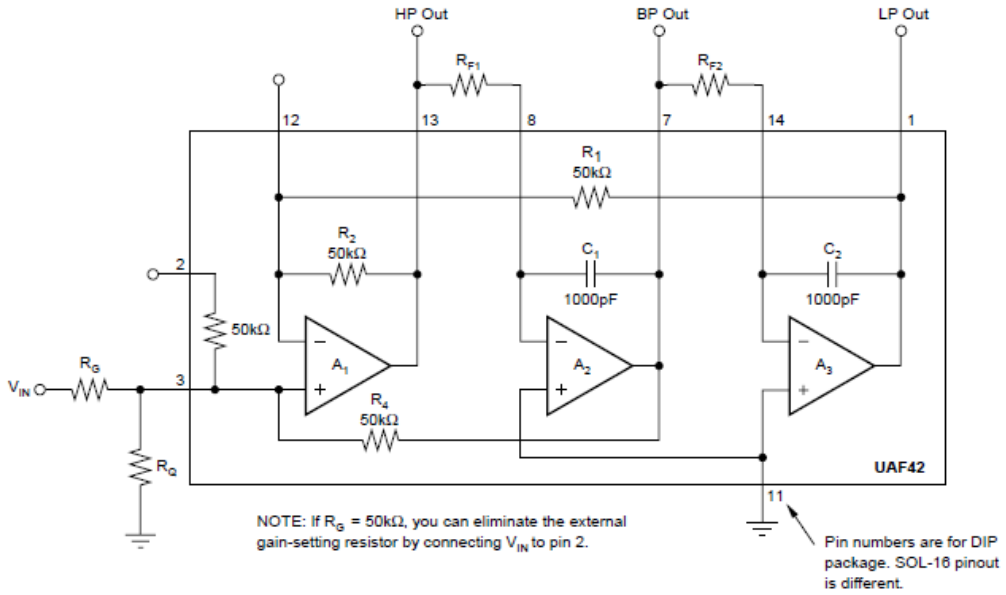


Figura 4.11 Circuito integrado del UAF42.

La estructura elegida para acoplar el UAF42 a nuestro circuito, es la siguiente:



Design Equations

1. $\omega_n^2 = \dfrac{R_2}{R_1 R_{F1} R_{F2} C_1 C_2}$

2. $Q = \dfrac{1 + \dfrac{R_4 (R_G + R_Q)}{R_G R_Q}}{1 + \dfrac{R_2}{R_1}} \left(\dfrac{R_2 R_{F1} C_1}{R_1 R_{F2} C_2} \right)^{1/2}$

3. $QA_{LP} = QA_{HP} \left(\dfrac{R_1}{R_2} \right) = A_{BP} \left(\dfrac{R_1 R_{F1} C_1}{R_2 R_{F2} C_2} \right)^{1/2}$

4. $A_{LP} = \dfrac{1 + \dfrac{R_1}{R_2}}{R_G \left(\dfrac{1}{R_G} + \dfrac{1}{R_Q} + \dfrac{1}{R_4} \right)}$

5. $A_{HP} = \dfrac{R_2}{R_1} A_{LP} = \dfrac{1 + \dfrac{R_2}{R_1}}{R_G \left(\dfrac{1}{R_G} + \dfrac{1}{R_Q} + \dfrac{1}{R_4} \right)}$

6. $A_{BP} = \dfrac{R_4}{R_G}$

Figura 4.12: Estructura del UAF42 y ecuaciones designadas

De la formula número 1, de la figura 4.13, obtendremos el valor de Rf1:

f=100 (Hz). Rf1=Rf2. R1=R2=50(KΩ)

→ $(2\pi \times 100)^2 = \dfrac{50\times10^3}{(50\times10^3)\times(100\times10^3)^2\times RF1^2}$ →

RF1 = 1.591.549,431(Ω)

→Para el cálculo de RF2, hemos cogido el mismo valor que RF1 y lo hemos ido variando hasta conseguir que coincidiera nuestra salida (pin1) con el paso por cero de nuestra señal de entrada.

Una vez explicado la adaptación al UAF42 (divisor de tensión y resistencias necesarias para acoplar la señal de entrada al UAF42), la alimentación del UAF42 (+/- 15(v)) y los resultados teóricos (elección de la estructura del UAF42 Y cálculos de RF1 y RF2).

Vamos a ver los resultados experimentales, las salidas del filtro UAF42, que salen en nuestro circuito, y se verá cual es la señal que se elige y su explicación.

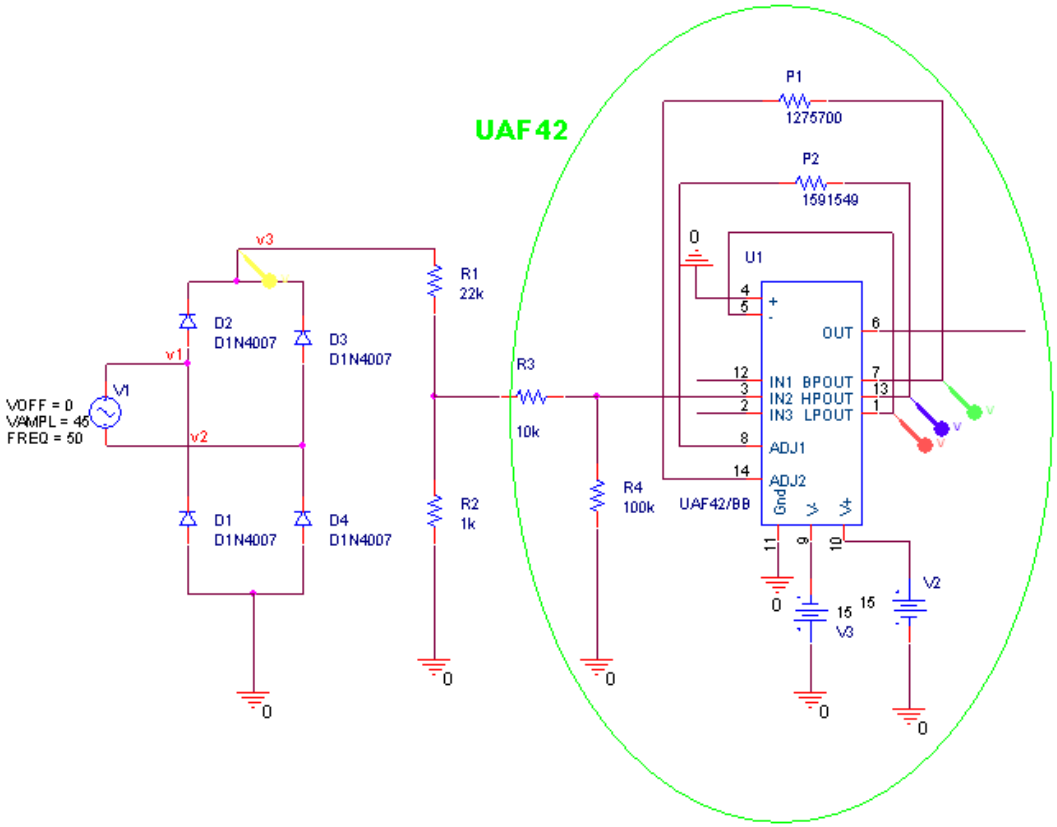


Figura 4.13 Conexión del UAF42

El resultado de las 3 señales del UAF42, de la figura 4.15, son las siguientes:

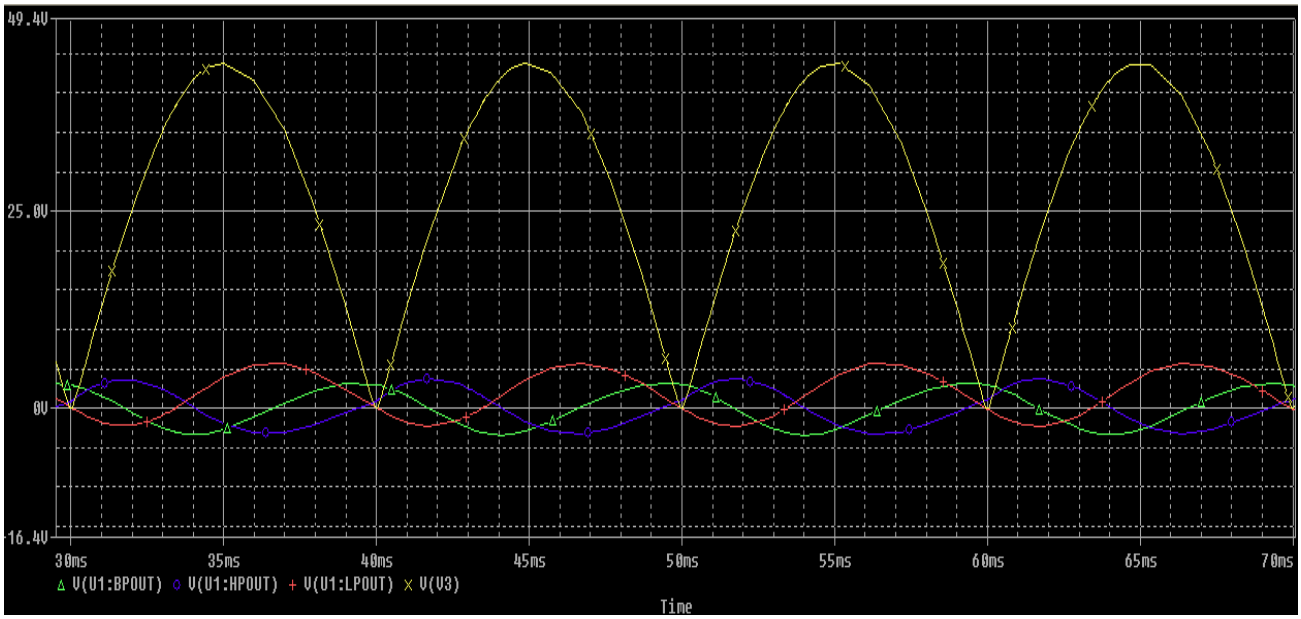


Figura 4.14: Señales de salida del UAF42: High-Pass; Band-Pass; Low-Pass.

Como podemos observar tenemos las 3 salidas del uaf42 (High-Pass; Band-Pass; Low-Pass), con una frecuencia de 100(Hz).

- HIGH-PASS (PIN 13), que corresponde con la señal de color azul, pero no coincide con el paso por cero de nuestra señal de entrada.

- BAND-PASS (PIN 7), que corresponde con la señal de color verde, que tampoco coincide con el paso por cero de nuestra señal de entrada.

- LOW-PASS (PIN 1)→ , que corresponde con la señal de color rojo, que realizando los cálculos pertinentes de Rf1 y Rf2 hemos conseguido que nuestra señal de salida del uaf42 (pin1) coincida con el paso por cero de nuestra señal de entrada.

Después de que la señal pase por el pin 1(Low-pass) (la cual es la elegida ya que su paso por cero coincide con nuestra señal de entrada), se vuelve a pasar por el comparador que contiene nuestro integrado UAF42, para conseguir una onda cuadrada:

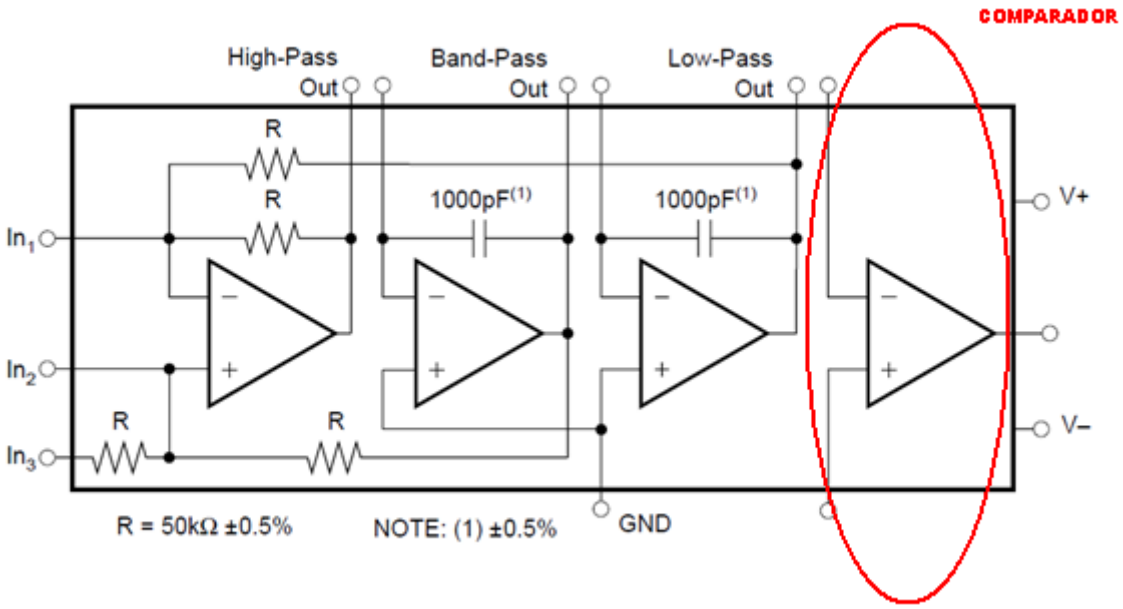


Figura 4.15: Comparador del UAF42

Cada vez que la tensión de entrada del comparador es <=0, el pulso de salida representa un pulso de nivel alto; si la tensión de entrada es >0, el pulso de salida representa un pulso de nivel bajo.

Vamos a ver en la figura 4.17, la representación de salida que hemos escogido del filtro UAF42, y la salida del comparador, que nos hemos ayudado del comparador interno que contiene el filtro. Comprobamos que, el periodo de la señal de salida del comparador, tiene un periodo de 10(ms), necesario para la sincronización con la red. Y, coincidente con el periodo de los instantes de conmutación, calculados en el apartado 3.

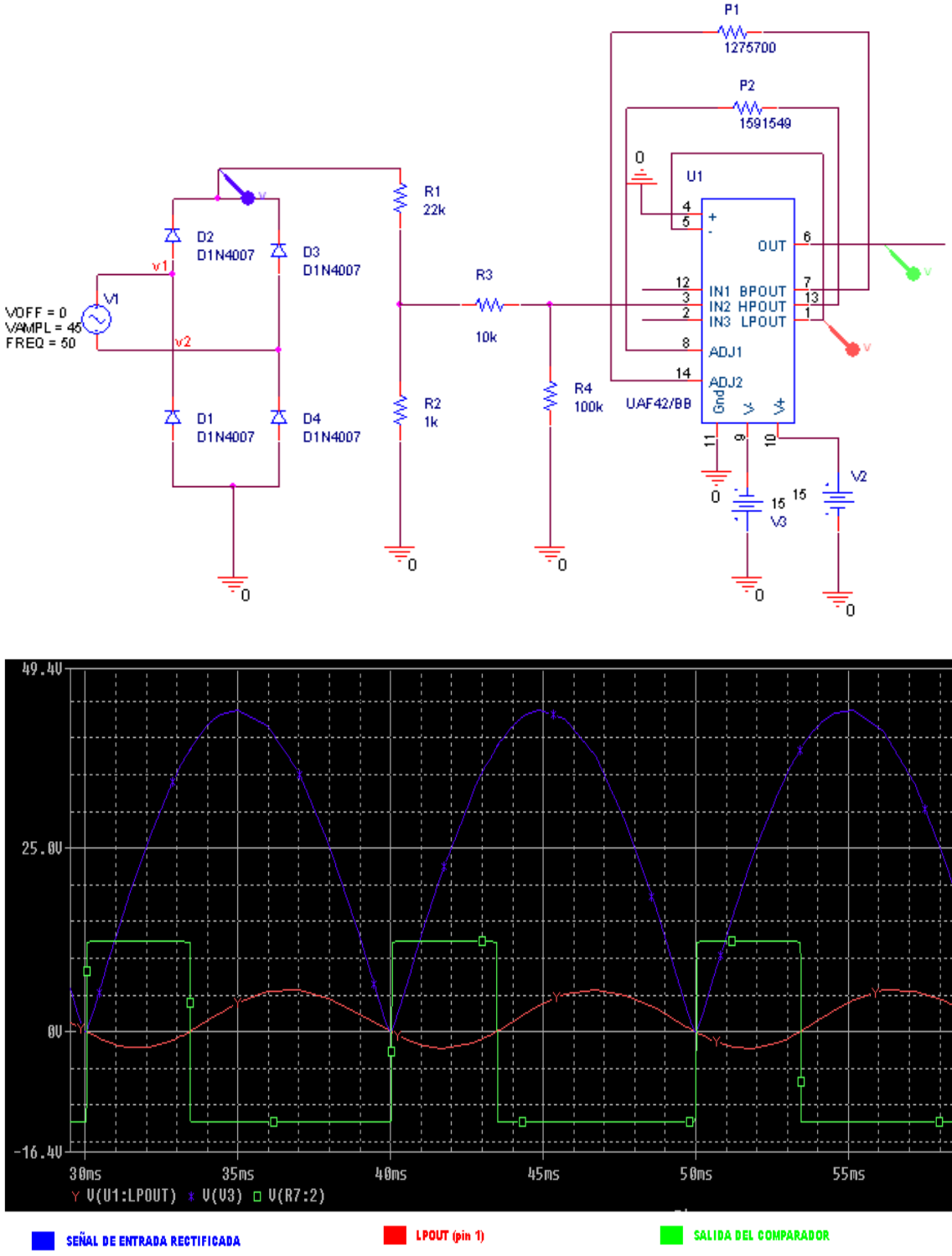


Figura 4.16: Salida del UAF42

Se puede comprobar como la señal de salida del comparador coincide con el paso por cero de nuestra señal de entrada rectificada, y que además el periodo de nuestra señal de salida es de 10(ms). La amplitud de salida del UAF42, se incrementa debido a la tensión de alimentación del integrado, que en este caso es de +/- 15(v), por lo que tendremos que poner un divisor de tensión para conseguir la señal de salida de 3.3 (v).A parte se tiene que llevar la señal de salida a la parte positiva, para ello se colocara un diodo.

La amplitud de la señal de salida es de 12(v), se realiza un divisor de tensión para reducirla a 3.3 (v):

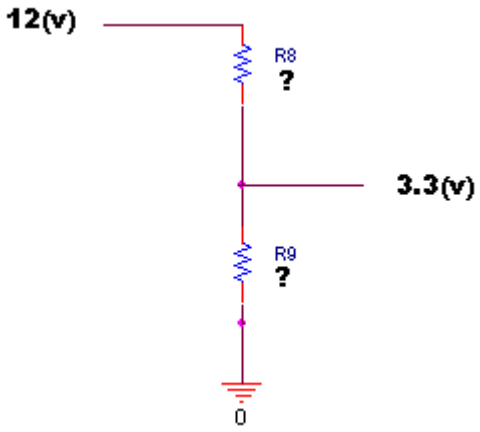
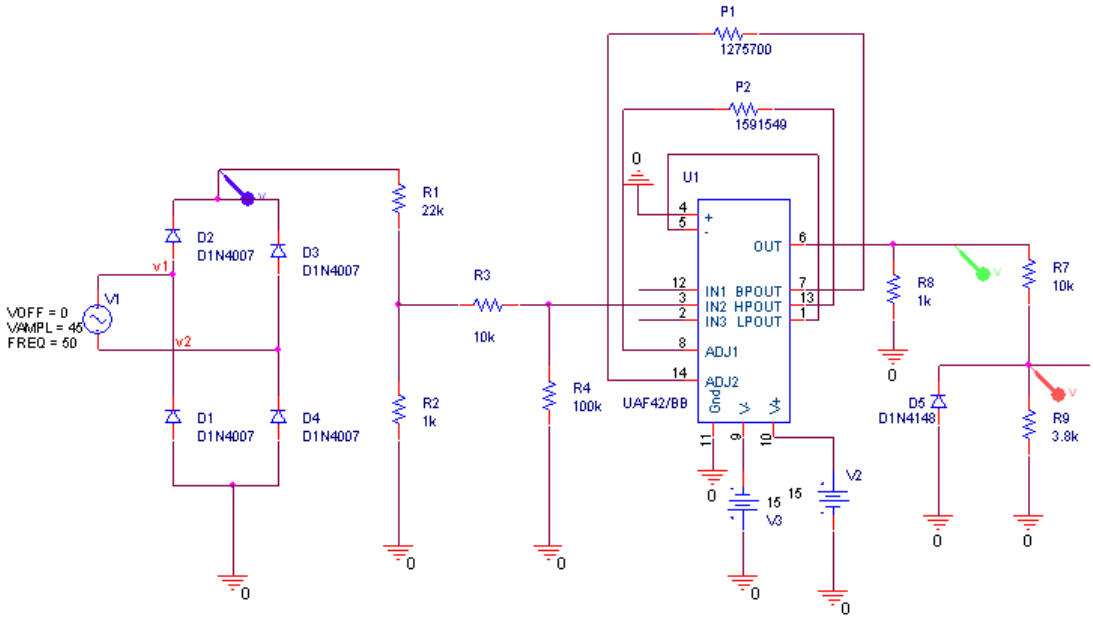


Figura 4.17: Divisor de tensión antes del 74HC14

$V_O = \frac{R_9}{R_8+R_9} \times V_{IN}$ → $3.3 = \frac{R_9}{R_8+R_9} \times 12$ →Le damos un valor a R_8:

→$R_8 = 10(K\Omega)$; Despejando nos sale que $R_9 = 3.79 \sim 3.8(K\Omega)$



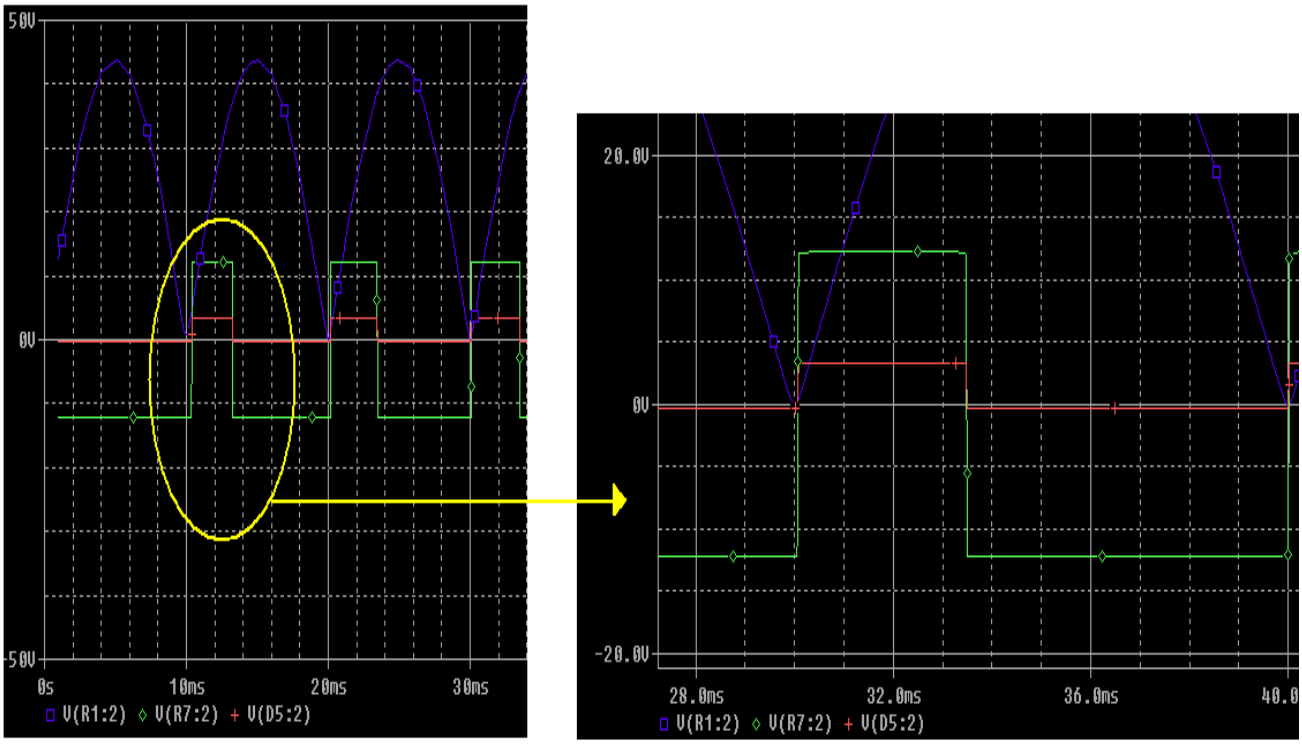
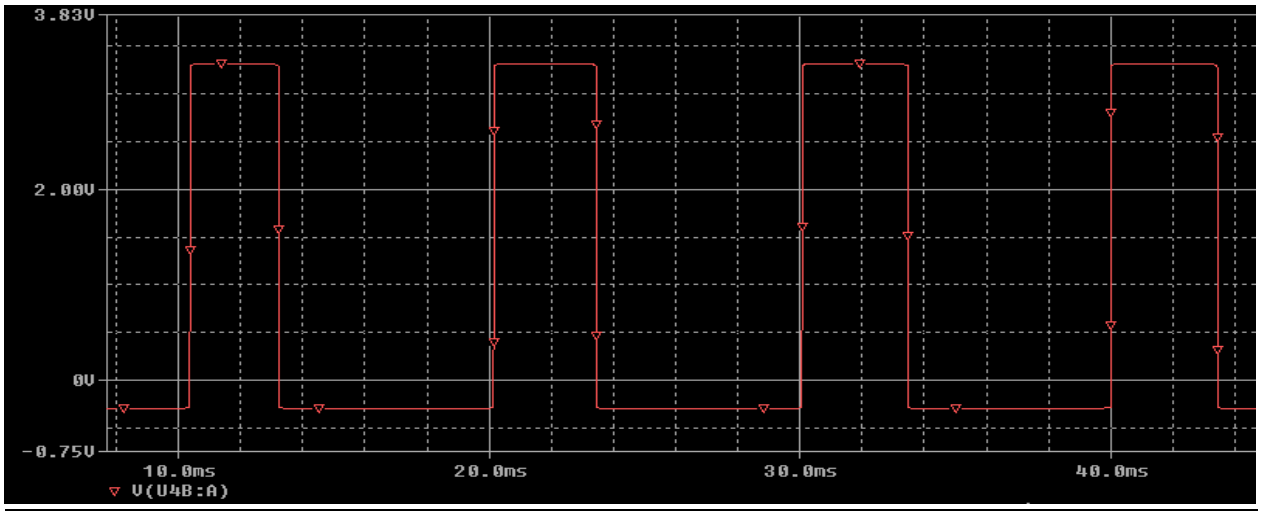


Figura 4.18: Acondicionamiento de la señal de salida del UAF42 (teórico)

A partir de ahora vamos a seguir explicando el circuito con los resultados experimentales, ya que en ORCAD no disponemos del integrado 74hc14, el cual, se compone de unas puertas inversoras con histéresis, cuyo fin, es llevar la señal a la parte positiva, ya que con el diodo no se consigue totalmente, que empiece en el origen, debido a su tensión. Solo se dispone en ORCAD, de puertas inversoras del integrado, pero de forma independiente. Vamos a ver si coincide la señal teórica de la figura 4.26, después del divisor de tensión de la señal de salida del uaf42, con la señal experimental:

<u>Señal teórica, resultante después del divisor de tensión, de la señal de salida del filtro</u>



Señal experimental, resultante después del divisor de tensión, de la señal de salida del filtro

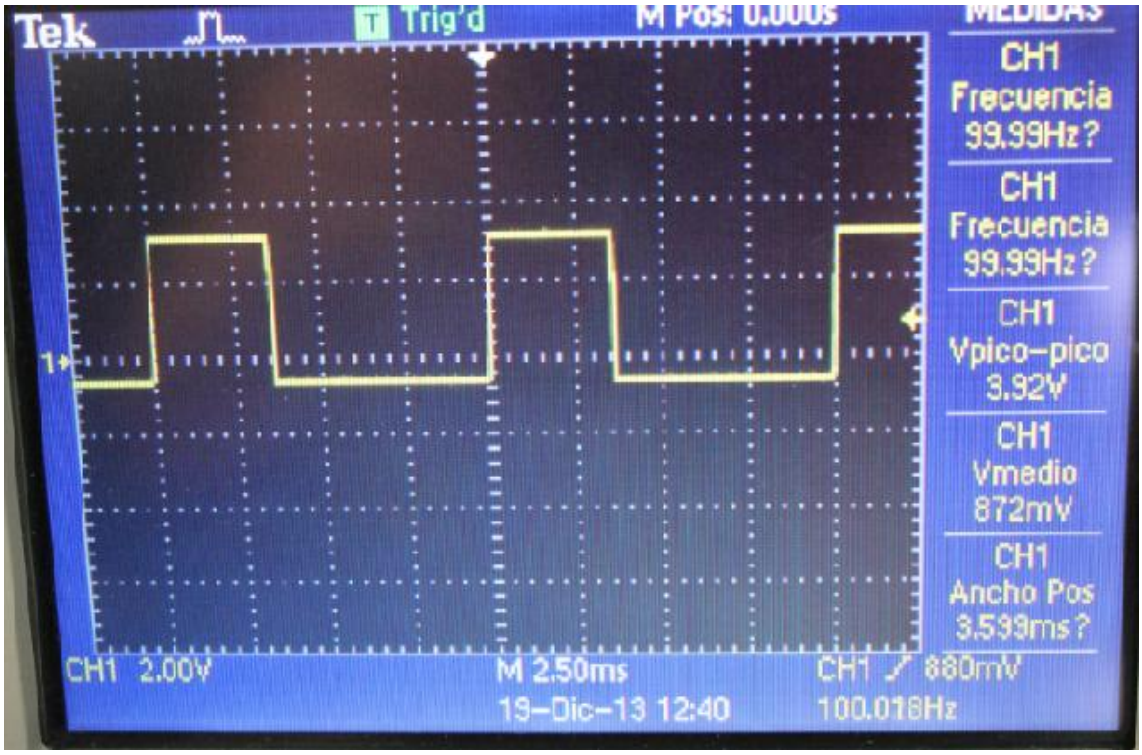


Figura 4.19: Acondicionamiento de la señal de salida del UAF42 (practico)

Observamos que la señal resultante tanto teórica como práctica, coincide.

Se obtiene la señal de 3,3(v), sincronizada con la red, pero tenemos que eliminar la parte negativa de la señal, por lo que, se pasara la señal por unas puertas inversoras con histéresis→ 74hc14.

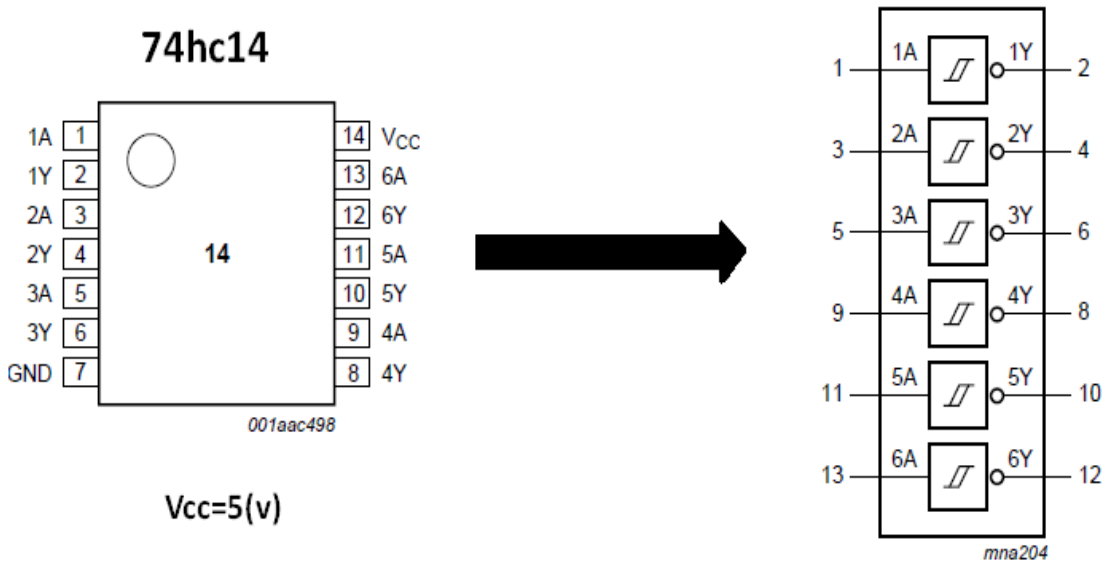


Figura 4.20: 74hc14

Se pasa la señal por dos puertas (PIN 1-2 y PIN 3-4), para recuperar su forma inicial y así, obtener nuestra señal sin la parte negativa. Pero al estar alimentado el integrado a 5(v), la amplitud de la señal ha crecido, por lo que tenemos que reducirla, mediante un divisor de tensión.

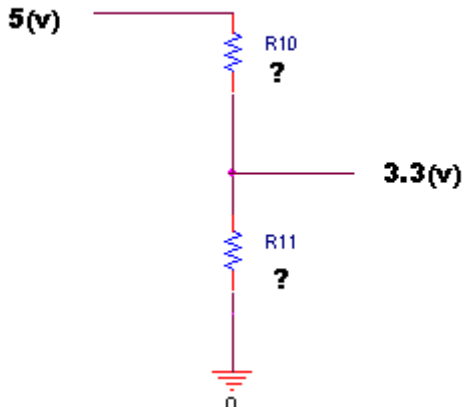


Figura 4.21: Divisor de tensión después del 74H14

$V_O = \frac{R_{11}}{R_{10}+R_{11}} \times V_{IN}$ → $3.3 = \frac{R_{11}}{R_{10}+R_{11}} \times 5$ →Le damos un valor a R_{10}:

→$R_{10} = 10(K\Omega)$; Despejando nos sale que $R_{11} = 19.4(K\Omega)$.

Finalmente se consigue la señal cuadrada, de amplitud 3.3 (v) y sincronizada con la red. Vamos a ver gráficamente como queda definitivamente nuestro circuito, en el siguiente apartado.

4.3 SEÑAL RESULTANTE DE NUESTRO CIRCUITO:

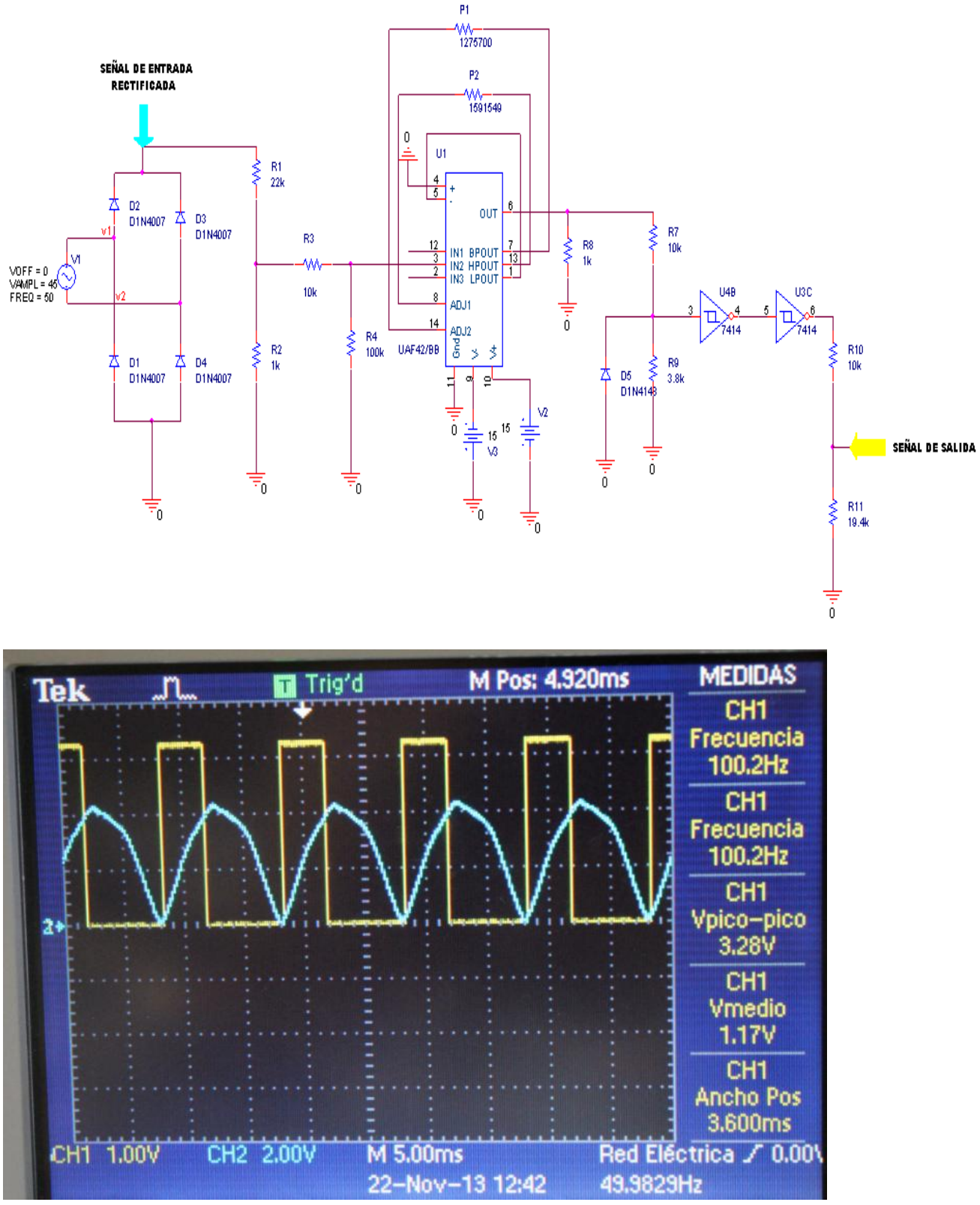


Figura 4.22 Señal de salida sincronizada con la señal de entrada rectificada

CONCLUSIONES

Vemos que el objetivo los hemos conseguido: una señal cuadrada de 3.3 (v) de amplitud, y esta sincronizada con la señal de entrada. Se necesita conseguir 3.3 (v) de salida de circuito, ya que esta señal se manda a la FPGA, y esta funciona con esa tensión.

A la hora de realizar las pruebas experimentales ya con todas las partes del circuito juntas, y realizando estas pruebas con otra fuente sinusoidal, de mayor tensión, se piensa, que se puede producir un desfase de las señales de entrada y de salida, del circuito (por la tensión de OFFSET de la nueva fuente), y que estas no coincidan en el paso por cero. Para evitar este posible problema, se coloca unos condensadores de 1(µF) a la entrada de la señal sinusoidal, para eliminar esa tensión de OFSSET de la fuente:

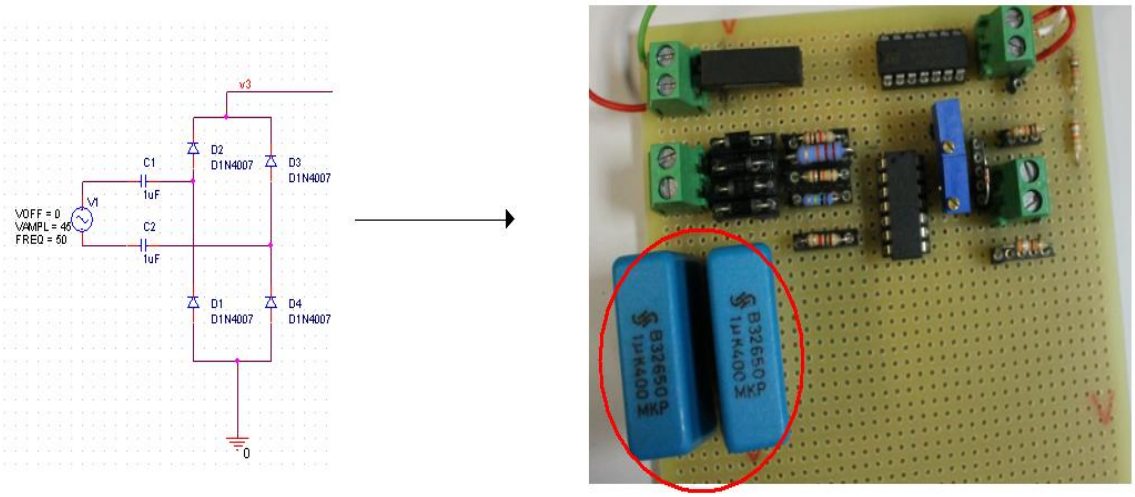


Figura 4.23: Eliminación de OFFSET

5 IMPLEMENTACIÓN Y CARACTERIZACIÓN DEL SISTEMA DE CONTROL EN LA FPGA

5.1) CARACTERISTICAS DE LA FPGA

LA FPGA:

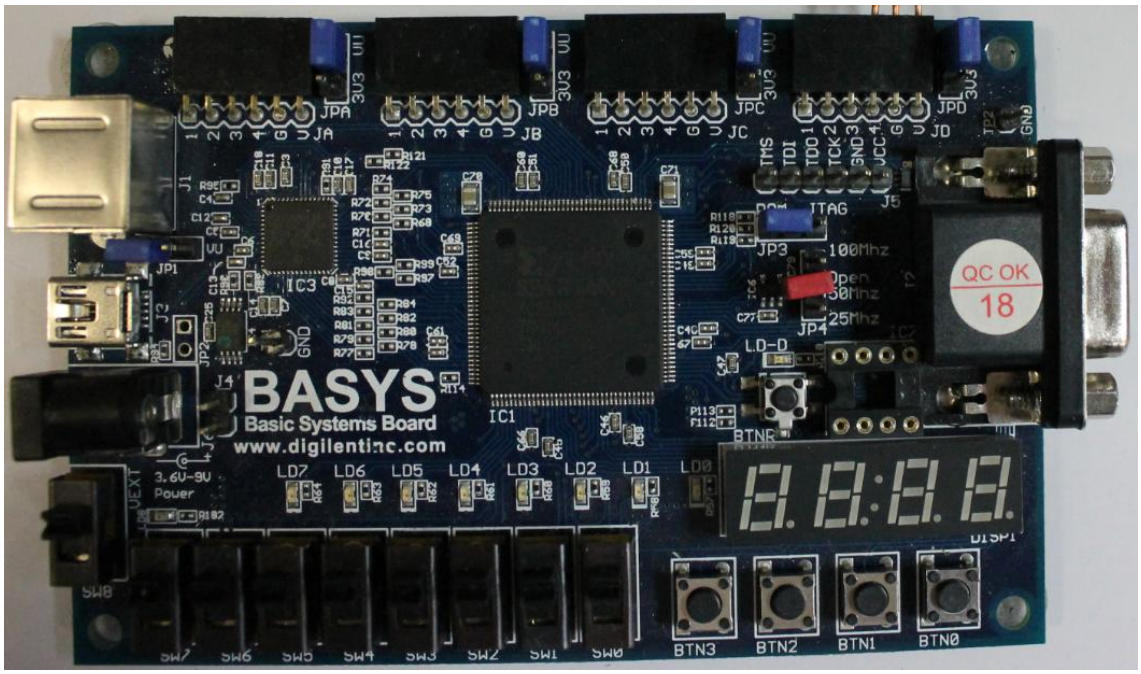


Figura 5.1 FPGA

PLACA FPGA:

Una FPGA (Field Programmable Gate Arrays) es un dispositivo lógico programable por el usuario, compuesto por bloques configurables comunicados entre sí. Por lo tanto, lo que programamos en una FPGA son los conmutadores, que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques.

La FPGA, pueden ser reprogramada (aunque en función de la tecnología utilizada para su implementación, hay algunos tipos que sólo pueden programarse una única vez). En la mayoría de las FGPA, la configuración es volátil (característica que también depende de la tecnología de implementación) por lo que si hay un corte de energía se tiene que volver a reprogramar.

Existen varios tipos de FPGA en el mercado, que difieren principalmente en su arquitectura interna, lo que también define su programación y eficiencia (retardos, área/volumen y costos).

El diagrama de bloques de nuestra FPGA es el siguiente:

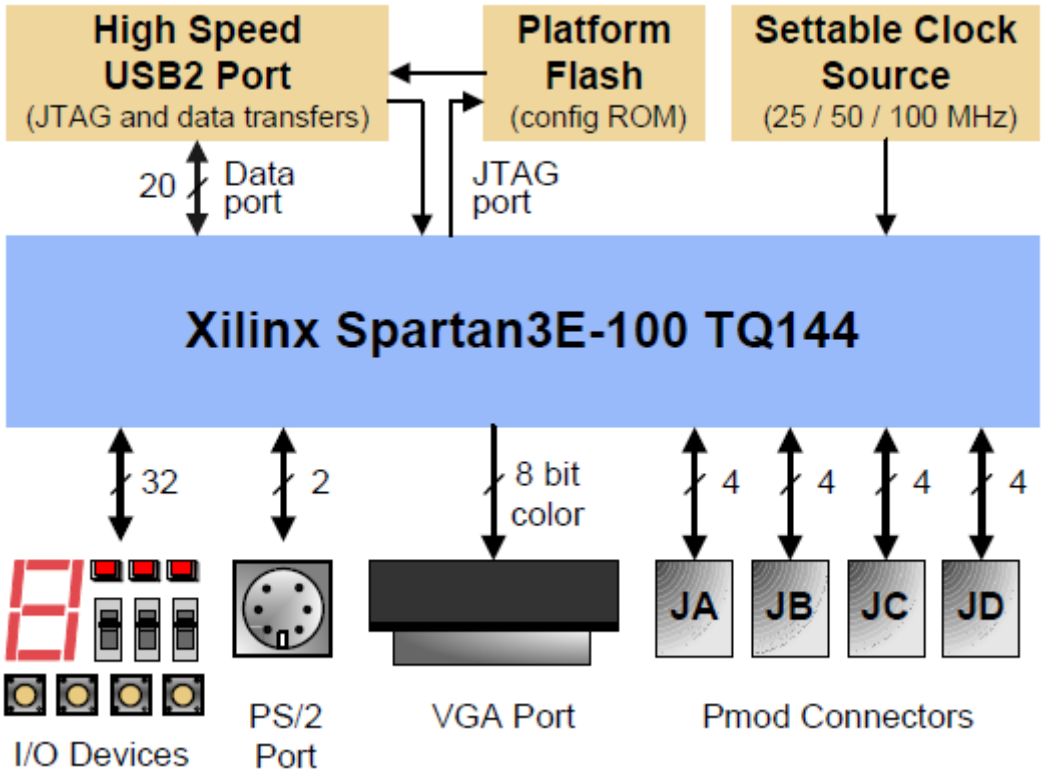


Figura 5.2: Diagrama de bloques de las partes de la FPGA

De los distintos dispositivos e interfaces disponibles, en la FPGA, se eligen los siguientes elementos para la implementacion del sistema:

- Se utilizara un interruptor, que actuara como Reset, para activar o desactivar el programa de la FPGA.

- De las frecuencias que proporciona el reloj de la tarjeta, se eligen 50MHz.

- De los conectores: utilizaremos 2 conectores, para la alimentacion de nuestra PCB, ya que la FPGA, nos proporciona una fuente de alimentacion continua. Además, se utilizará 1 conector para la entrada, de la señal de nuestro circuito("syncro"). Otros, 5 conectores para,la salida de los 5 modulos, sincronizados con la red.

- Se utilizara el puerto USB para comunicarse con la FPGA y enviar el codigo de nuestro circuito digital.

- Se programa inicialmente la propia FPGA para ir probando nuestro circuito digital. Cuando ya se consigue que funcione correctamente, se pasa a la memoria asociada a la FPGA de manera que no sea necesario configurar el sistema cada ver que se enciende la tarjeta.

ENTRADAS Y SALIDAS de la FPGA:

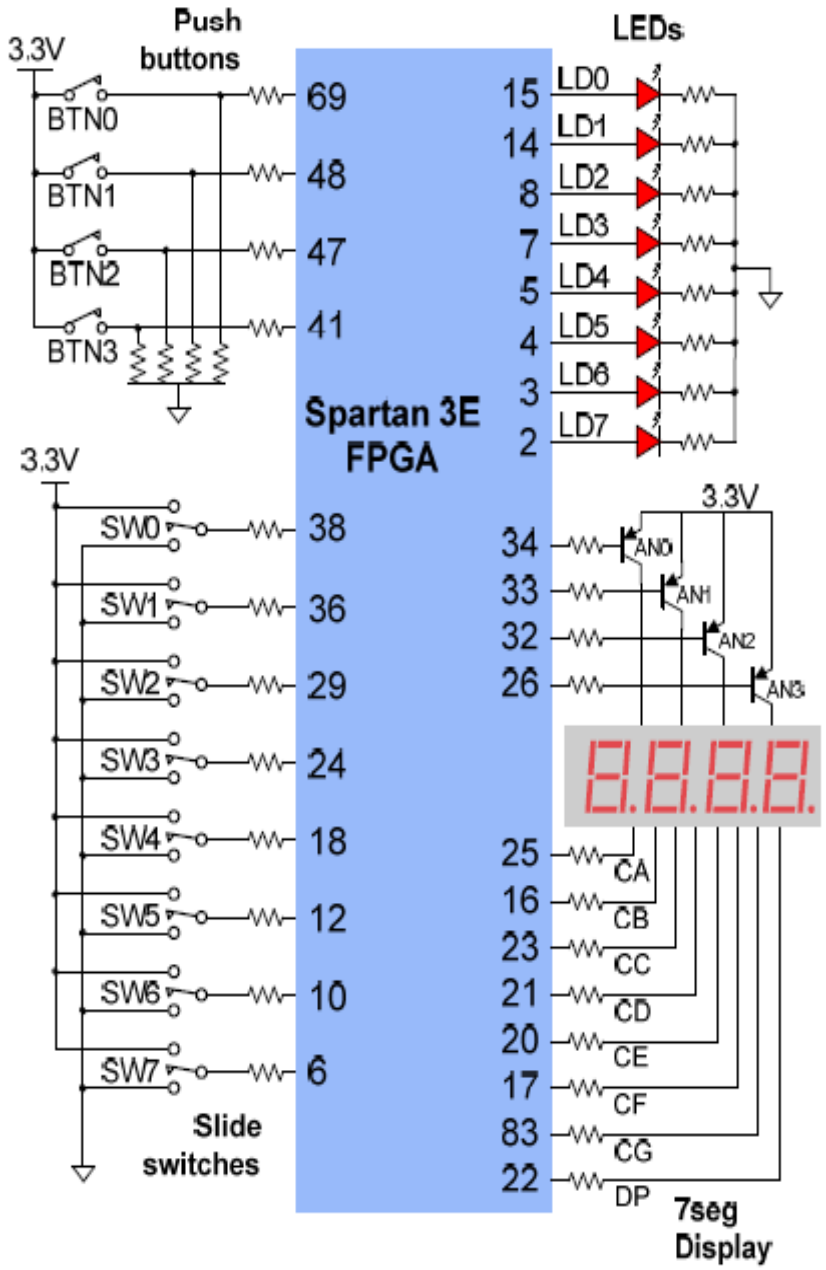


Figura 5.3: Distribución de las entradas y salidas

Distribución de las entradas y salidas de la FPGA:

Cuatro pulsadores y ocho interruptores deslizantes, nos proporcionan las entradas del circuito. Los pulsadores de entrada, están a nivel bajo('0'), que se activaran a nivel alto('1'), sólo cuando se pulsa el botón pulsador. Interruptores deslizantes, se activan a nivel alto o a nivel bajo, dependiendo de la posición del interruptor. Botones, pulsadores e interruptores deslizantes tienen resistencias en serie, para la protección contra cortocircuitos (se produciría un cortocircuito si un pin de la FPGA, asignado a un pulsador o interruptor deslizante, se define indebidamente como una salida).

Ocho LEDs y una Pantalla LED, de cuatro dígitos y siete segmentos, se proporcionan para las salidas del circuito. El ánodo de los LED, son accionados desde la FPGA, a través de limitador de corriente, por lo que se encienden, cuando un '1' lógico se escribe en el correspondiente pin de la FPGA. Un noveno LED se proporciona, como un LED indicador de alimentación, y el LED décimo (LD- D) se ilumina cada vez que la FPGA ha sido programada con éxito.

Display de siete segmentos:

Cada uno, de los cuatro dígitos, de 7 segmentos de la Pantalla LED, se compone de siete segmentos LED, dispuestos en un patrón de "figura 8". Los segmentos LED pueden ser iluminados de forma individual, por lo que cualquiera de los 128 patrones, se pueden mostrar en un dígito, al iluminar ciertos segmentos LED y dejando los demás apagados. De estos 128 patrones posibles, los diez que corresponden a los dígitos decimales son los más útiles.
Los ánodos de los siete LEDs, que forman cada dígito están unidos en un solo nodo del circuito de ánodo común, pero los cátodos LED permanecen separados.

CONEXIÓN DE LA FPGA

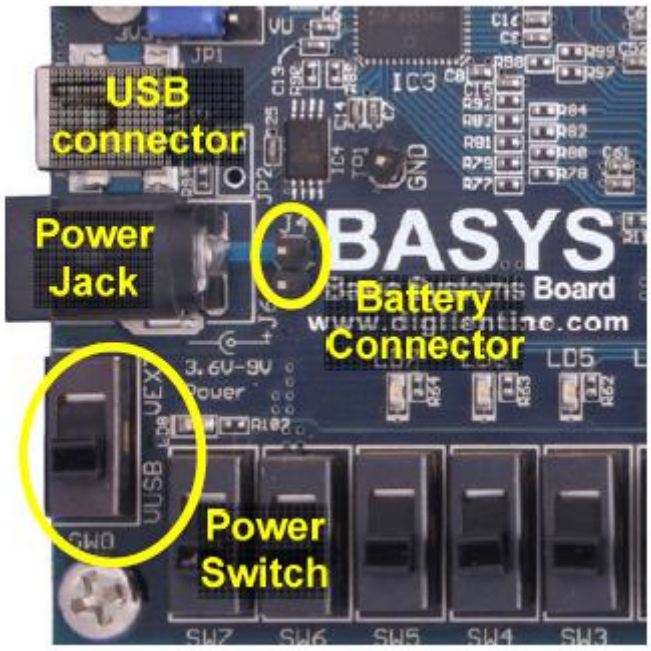


Figura 5.4: Conexiones de la FPGA

La placa Basys puede ser alimentado por un cable USB, por lo que el interruptor (SW8), tendría que estar en la posición USB; y si es alimentado por una batería externa, la posición del interruptor (SW8), tendría que estar en VEXT, esta última es la posición que nosotros tenemos en nuestra placa, ya que la hemos alimentado con una fuente de 5(v).

La batería, que se puede conectar a nuestra FPGA, tiene que estar dentro de un rango de entre 4(v) y 9 (v).

NOTA: Nuestra FPGA dispone de una memoria interna:

- A la hora de ir haciendo pruebas, hasta conseguir el código correcto del circuito digital, se envía la programación a la FPGA, mediante el programa DIGILENT ADEPT.

- En el instante, en el que ya tenemos el código de nuestro circuito digital final y todo funcione correctamente, introducimos nuestra programación en la memoria interna de la FPGA.

Distribución de los módulos:

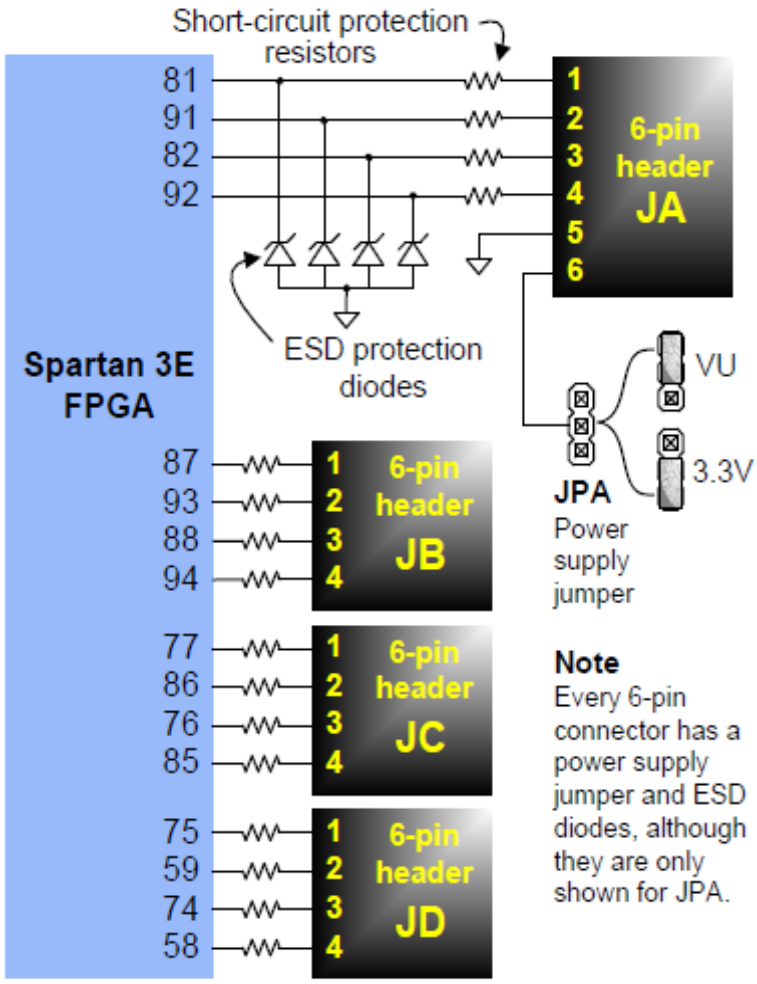


Figura 5.5: Elección de los pines, para las señales de salida de nuestros 5 módulos

Se cogen 5 pines, como salidas de nuestra FPGA, de las varias que existen, como se puede observar en la figura 5.5. En estos pines elegidos, podremos visualizar la distribución de cada módulo.

MODULO1→ PIN 81

MODULO2→PIN 82

MODULO3→PIN 87

MODULO4→PIN 93

MODULO5→PIN 88

PINES

En la tabla, mostramos la distribución de cada pin de la FPGA. Los pines que aparecen en gris no están disponibles para el usuario.

Basys Spartan-3E pin definitions											
Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	PROG	25	CA	49	VDDO-2	73	GND	97	PS2D	121	VDDO-0
2	LD7	26	AN3	50	GRN2	74	JD-3	98	NC	122	U-INT0
3	LD6	27	GND	51	GRN1	75	JD-1	99	GND	123	U-FLAGC
4	LD5	28	VDDO-3	52	GRN0	76	JC-3	100	VDDO-1	124	U-FLAGB
5	LD4	29	SW2	53	CLK2	77	JC-1	101	NC	125	U-FLAGA
6	SW7	30	VDDAUX	54	CLK1	78	NC	102	VDDAUX	126	U-IFCLK
7	LD3	31	NC	55	GND	79	VDDO-1	103	NC	127	GND
8	LD2	32	AN2	56	NC	80	VDDINT	104	U-SLWR	128	NC
9	VDDINT	33	AN1	57	MODE2	81	JA-1	105	U-SLRD	129	NC
10	SW6	34	AN0	58	JD-4	82	JA-3	106	U-SLCS	130	U-D7
11	GND	35	VS	59	JD-2	83	CG	107	NC	131	U-D6
12	SW5	36	SW1	60	MODE1	84	NC	108	TMS	132	U-D5
13	VDDO-3	37	GND	61	GND	85	JC-4	109	TDO	133	GND
14	LD1	38	SW0	62	MODE0	86	JC-2	110	TCK	134	U-D4
15	LD0	39	HS	63	DIN	87	JB-1	111	NC	135	U-D3
16	CB	40	INIT	64	VDDO-2	88	JB-3	112	U-PKTD	136	NC
17	CF	41	BTN3	65	VDDAUX	89	NC	113	U-FAD1	137	VDDAUX
18	SW4	42	VDDO-2	66	NC	90	GND	114	NC	138	VDDO-0
19	GND	43	BLUE1	67	RED2	91	JA-2	115	VDDINT	139	U-D2
20	CE	44	BLUE0	68	RED1	92	JA-4	116	U-FAD0	140	U-D1
21	CD	45	VDDINT	69	BTN0	93	JB-2	117	U-SLDE	141	NC
22	DP	46	GND	70	RED0	94	JB-4	118	GND	142	U-D0
23	CC	47	BTN2	71	CCLK	95	NC	119	NC	143	HSWAP
24	SW3	48	BTN1	72	DONE	96	PS2C	120	NC	144	TDI

FPGA pin definition table color key		
Grey		Not available to user
Green		User I/O devices
Yellow		Data ports
Tan		Pmod connector signals
Blue		USB signals

Figura 5.6: Tabla de los pines de la FPGA

FRECUENCIA ELEGIDA:

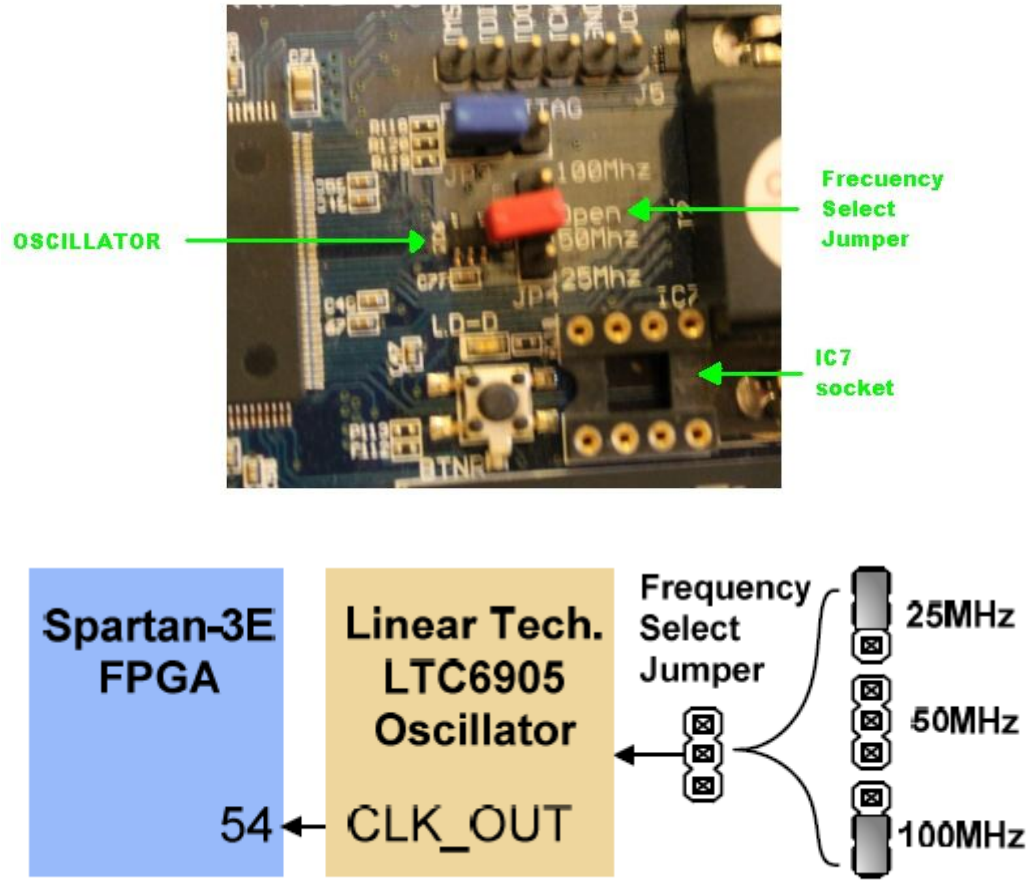


Figura 5.7: Selección de frecuencia elegida en la FPGA

La frecuencia seleccionada es de 50MHz, por lo que nuestro periodo:

$$Tclk = \frac{1}{50\times10^{6}} = 0{,}2\ \times 10^{-7}(segundos);$$

RESET:

Para resetear el programa, hemos escogido uno de los 7 interruptores deslizantes de los que dispone la FPGA→ (SW1).

Figura 5.8: Interruptores deslizantes de la FPGA

Mientras sea 0 el RESET no se ejecutara el código, hasta que no pongamos a 1 el interruptor.

ENTRADAS Y SALIDA DE LA FPGA:

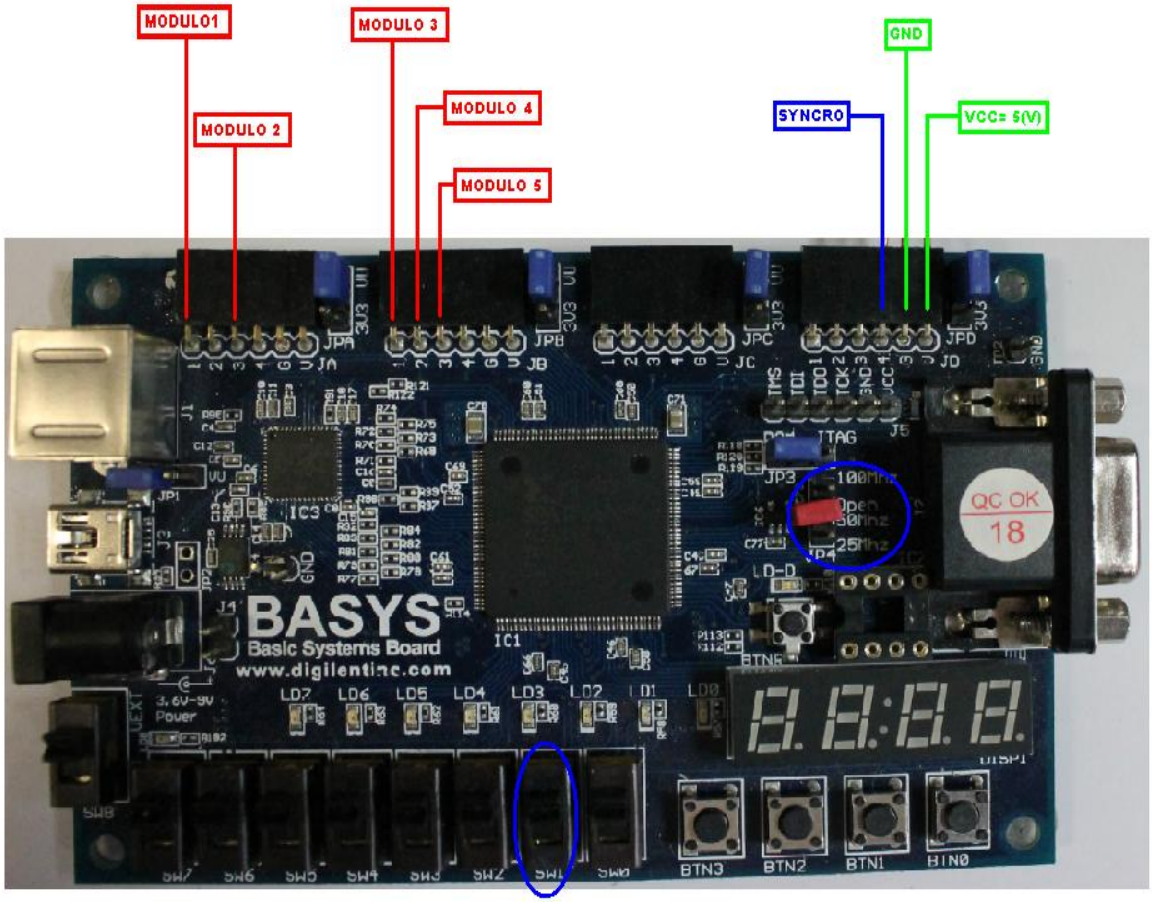


Figura 5.9: Elección de los pines utilizados de la FPGA

SALIDAS

ENTRADAS

VCC Y GND

→Las salidas de nuestra FPGA corresponden con la visualización de los 5 módulos.

→La entrada de nuestra FPGA:

- la señal SYNCRO, que es la señal de salida de nuestro circuito.
- CLK Y RESET, como ya se ha explicado anteriormente.

→VCC y GND: nos las genera nuestra FPGA, que se utilizara para alimentar nuestro circuito.

CONEXIONES DE LA FPGA Y LA PLACA

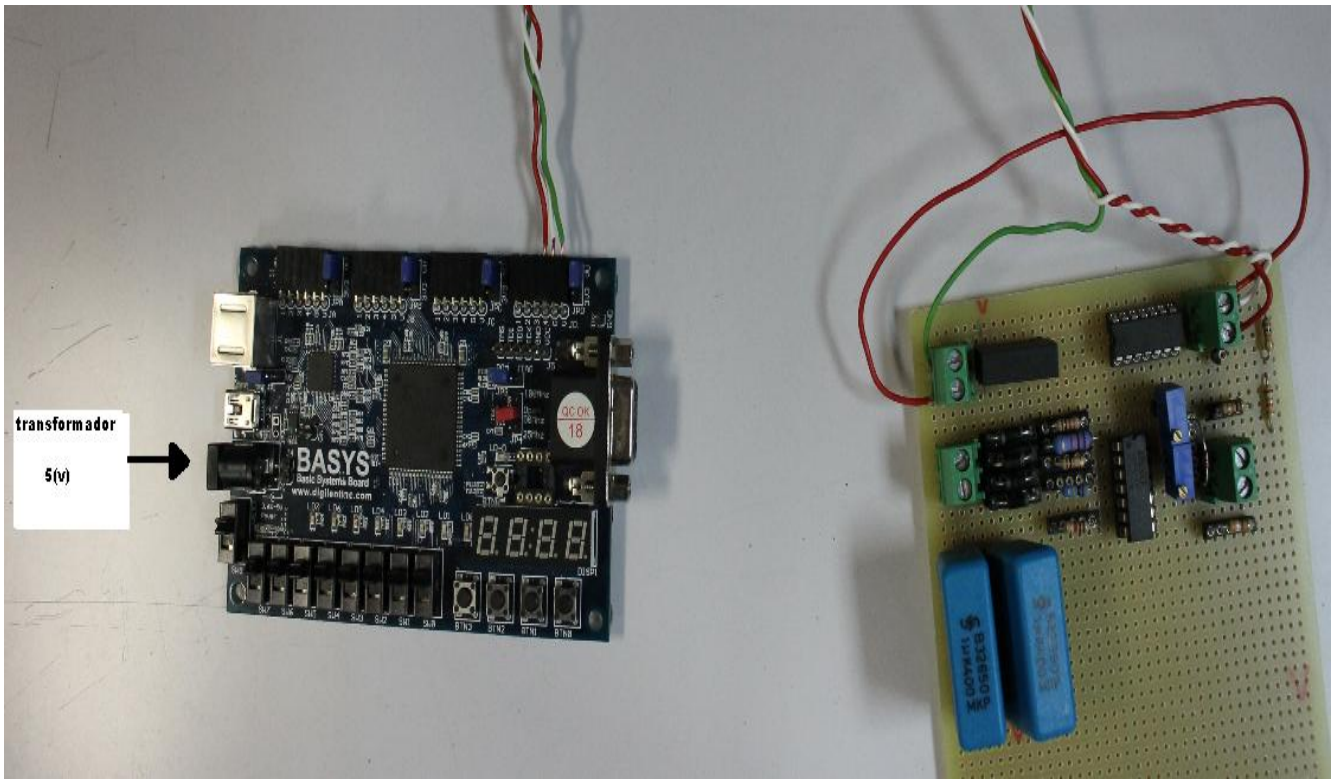


Figura 5.10: Pin de la FPGA que nos proporciona la alimentación continúa de nuestro circuito

La alimentación sale de la patilla → "V" (cable verde) y "GND" (cable blanco), de la FPGA, ver figura 5.10. La alimentación continua de la placa, la proporciona la FPGA, la cual, se alimenta con una fuente de 5(V). Esta alimentación, servirá para pasarla por el convertidor y conseguir +/- 15(V), para alimentar el filtro.

Por el cable rojo sale la señal "syncro", de la placa a la FPGA, como se puede observar en la figura 5.10

NOTA:
Con esta fuente, alimentamos la FPGA:

Figura 5.11: Fuente continua que alimenta el circuito

5.2) PROGRAMACION Y SINCRONIZACION DE LA FPGA CON LA SEÑAL SYNCRO.

El objetivo de nuestro programa, es conseguir sincronizar nuestra señal resultante del circuito (señal syncro), con el disparo de cada módulo (encendido y apagado de los 5 módulos, que corresponden con los datos de las tablas extraídos de matlab).

A continuación, se explica en líneas generales, las líneas de código que hemos utilizado para conseguir nuestra finalidad.

Nuestras entradas y salidas:

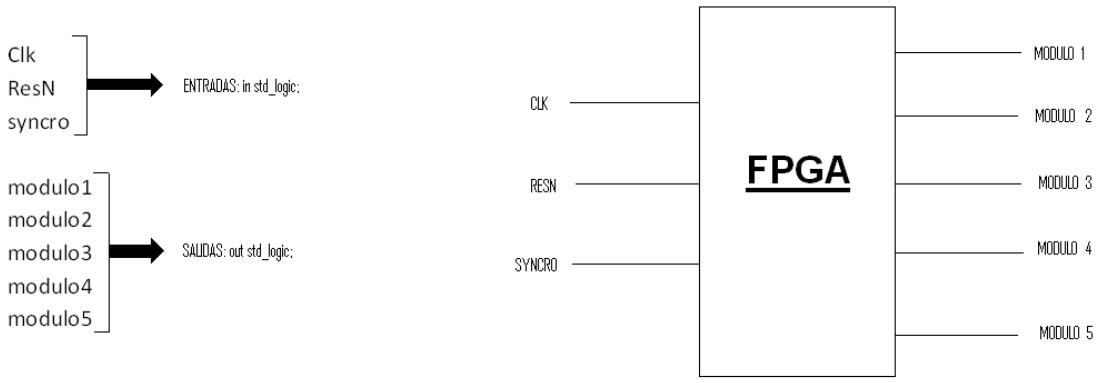


Figura 5.12: Diagrama de las entradas y salidas de la FPGA

CLK:

Tenemos que ajustar nuestro CLK, al de la FPGA (50 MHz), para ello se realiza el siguiente cálculo:

- ➔ $\frac{10(ms)}{10000} = 10 \times 10^{-7}$; se divide el periodo de nuestra señal entre los 10.000 puntos en los que se ha dividido el periodo. Tiempo que tiene que pasar antes de cada punto.
- ➔ $Tclk = \frac{1}{50 \times 10^6} = 0{,}2 \times 10^{-7}(segundos)$;

Multiplicamos el Tclk de nuestra FPGA y nuestro tiempo que tiene que pasar antes de cada punto, así conseguiremos el tiempo necesario que tiene que pasar para poder pasar de un punto a otro.

Sale que ay que esperar 50 eventos del CLK, antes de incrementar el tiempo y ver, si en ese instante, se produce un flanco de subida o de bajada en algún modulo.

RESET

Se ha utilizado, un interruptor deslizante de la FPGA como Reset, mientras sea 0 no se ejecutara el código, hasta que no se ponga a 1 el interruptor.

SYNCRO

Es la señal de salida de nuestro circuito, sincronizada con la señal de red, con un periodo de 10 ms.

MODULOS 1, 2, 3, 4 Y 5

Son las salidas de nuestra FPGA, se sacaran por varios pines de salida que se han elegido de la FPGA, en donde se puede visualizar los ángulos (encendido y apagado) de cada módulo.

A continuación explicaremos cada bloque, en los que se ha dividido el programa, para su correcto funcionamiento:

INSTANCIACION DE LAS TABLAS

Se crean 2 matrices, una con los instantes de tiempo, en los que se producen un cambio de los ángulos de cada módulo (matriz ángulo), y otra matriz que marcara si es un flanco de subida, con un 1, o de un flanco de bajada con un 0(matriz bit).

Lo que se consigue en este bloque, es traer a nuestro programa principal los valores de las tablas (ADDR, DOUT y LOUT), mediante el siguiente código:

```
ADDR=> ADDR_tabla

DOUT=> Data_IN_tabla

LOUT=> Nivel_IN_tabla
```

ADDR: es un puntero, para marcar la posición correspondiente, de la matriz ángulo y de la matriz bit, y así obtener el dato correcto, que pertenezca a la misma posición, en ambas matrices.

DOUT: es un numero entero de rango 0 a 10000, muestra el contenido de la matriz ángulo, cada vez que el puntero ADDR, marca una posición de esta tabla.

LOUT: es un vector de 5 posiciones (4 down to 0), muestra el contenido de la matriz ángulo, cada vez que el puntero ADDR, marca una posición de esta tabla.

CONTADOR PRINCIPAL

En este bloque, se utilizan 2 contadores: El primer contador que tenemos es “cont”, cuyo fin de cuenta lo marcamos con la señal “tpoen”. El contador “cont”, es necesario para acoplarse al CLK de nuestra FPGA, hasta que no llegue a la cuenta de 49, no se incrementara el próximo contador de “tiempo”.

El contador “tiempo”, mientras sea menor de 10.000 se ira incrementado, y cuando llegue a su fin de cuenta se activara la señal “en”. Este contador llega hasta los 10.000, porque hasta esa cifra llegan los instantes de tiempo de los módulos.

LECTURA TABLA

En este bloque tenemos un puntero (ADDR_tabla), que nos indica la posición correspondiente en ambas matrices. Si el puntero vale 0, este apuntara al primer dato de la matriz ángulo y al primer dato de la matriz bit. Estos datos lo mandamos a nuestro programa principal mediante el código descrito en instanciación de tablas. Los valores de ambas matrices los guardamos en otra variable, mediante el siguiente código:

Data_IN<= Data_IN_tabla

Nivel_IN<= Nivel_IN_tabla

Cuando Data_IN, coincide con el "tiempo", quiere decir que en ese momento se produce un flanco, y guardamos el valor de Nivel_IN, en un vector auxiliar: Nivel_IN_aux.

Cada vez que coincida el tiempo con Data_IN, incrementamos el puntero, hasta que el puntero alcance su valor máximo.

SINCRONIZACION

S1→señal de nuestro programa; es la misma señal syncro, pero retrasada un ciclo de reloj, para que en ese intervalo se produzca un pulso (synch_en→señal).

Synch_en <= syncro and (not s1);

Con esta señal conseguimos resetear el programa, y que empiece la cuenta de nuestro tiempo.

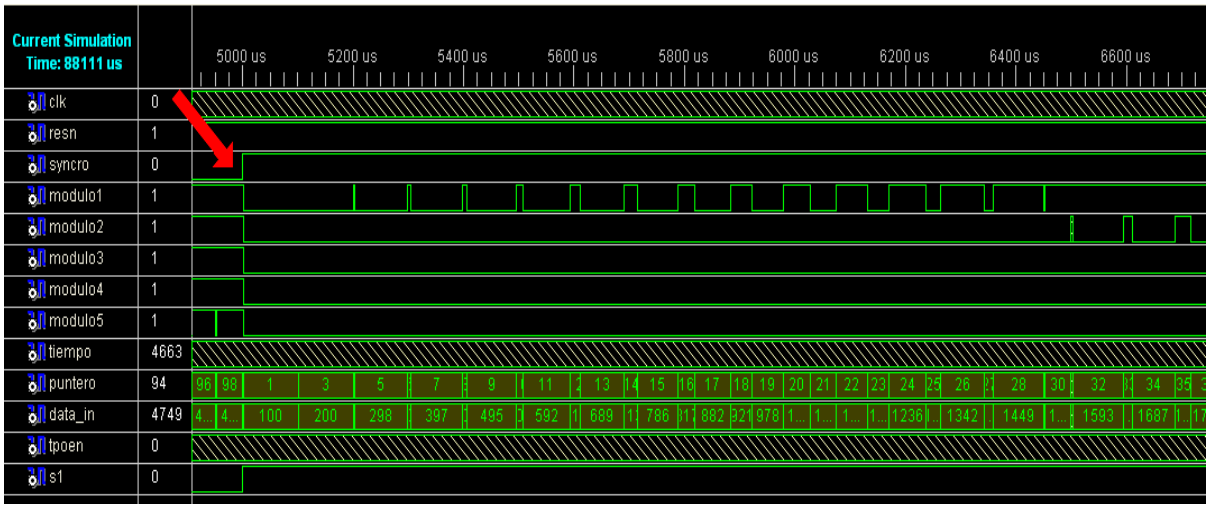


Figura 5.13: Simulación en XILINX del programa

VISUALIZACION DE LOS MODULOS

En este bloque, cuando se activa la señal "en" a 1, el cual se activa cuando "tiempo", ha finalizado su cuenta, enviamos los valores mediante un vector a los 5 módulos.

5.3) DIAGRAMA DE BLOQUES DEL PROGRAMA

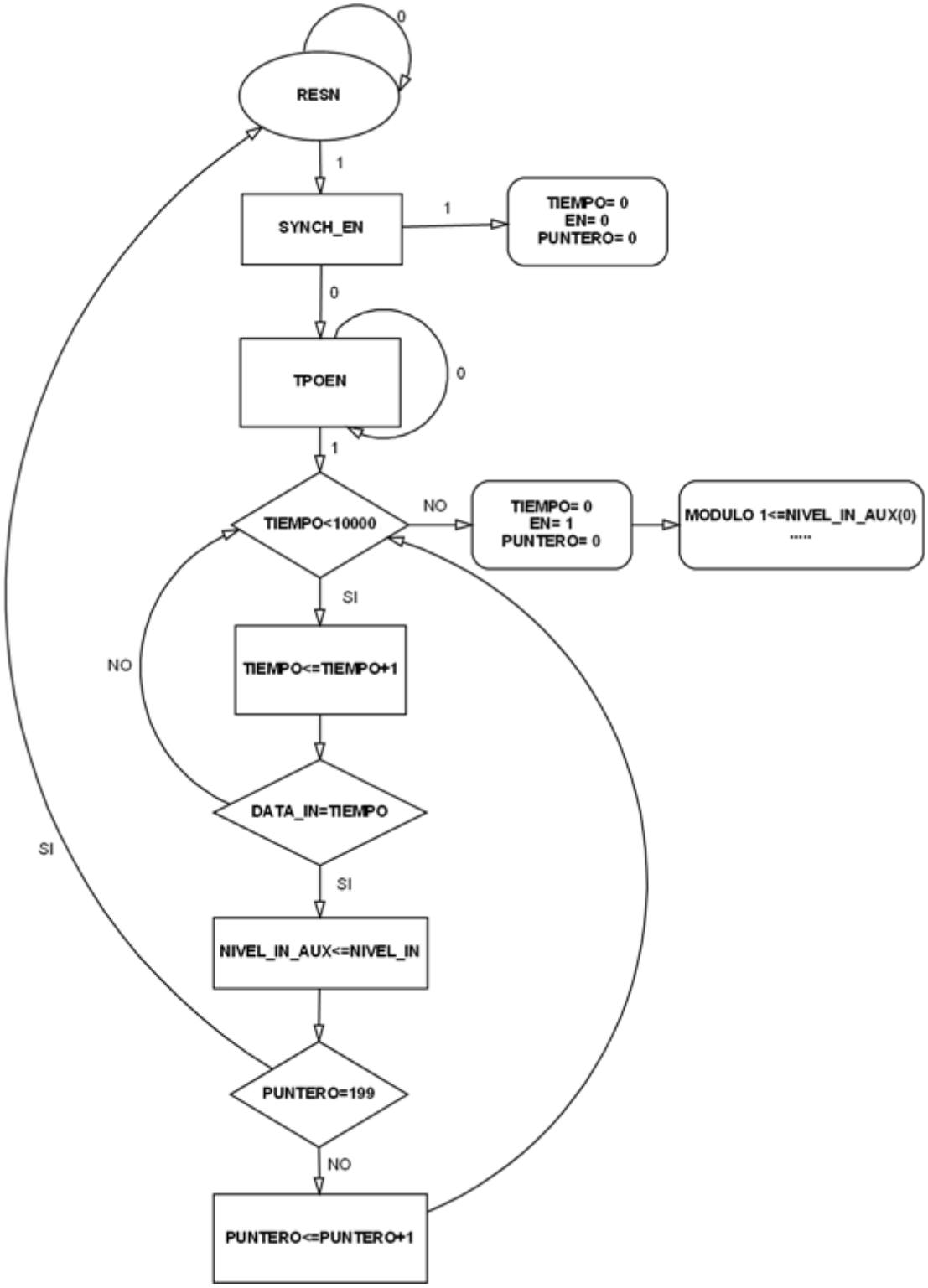


Figura 5.14: Diagrama de bloques del programa

Una vez obtenida, la señal de salida del circuito (syncro), las tablas obtenidas en MATLAB en las que queda marcado cada instante de conmutación de cada módulo, y que por ultimo hemos conseguido sincronizar estas dos partes mediante el programa que hemos realizado, ya

se puede realizar las pruebas necesarias para ver el funcionamiento correcto, como se puede ver en el siguiente capítulo.

Conclusiones:

La simulación en XILINX, de cómo quedarían las salidas de nuestro código seria:

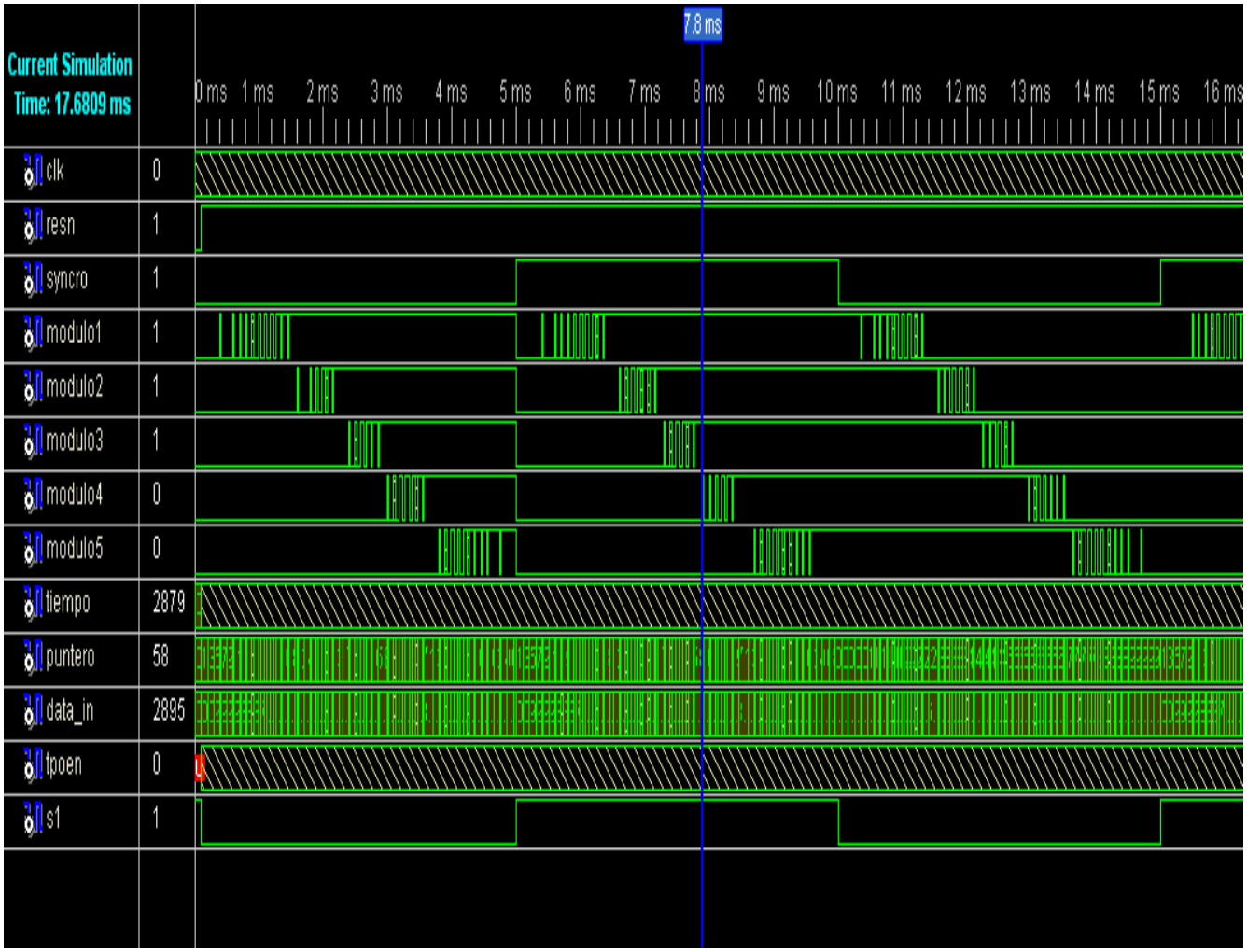


Figura 5.15: Representación en XILINX, de las salidas de la FPGA.

Se observa en la figura 5.15, la correcta sincronización de nuestra señal “syncro”, con los instantes de conmutación. Cuando se activa, el Reset(interruptor deslizante),a nivel alto, comienza el programa. Cuando la señal syncro, se activa, se resetea el programa y empieza desde el principio. Y, así se repetirá sucesivamente, asegurando la sincronización con los instantes de conmutación.

6 COMPROBACION DE RESULTADOS

Estos resultados se han realizado con una tensión de entrada de alterna de 45(V) de amplitud, para comprobar que el encendido y apagado de los led funciona correctamente, antes de probarlo directamente en el tubo de led, en la que se probara con otra fuente de alterna, con más tensión.

Una vez creado el programa que sincronice, la señal "syncro" (salida de la PCB) con los instantes de conmutación, solo queda mandar el código del PC, a la FPGA, e ir visualizando que todos los ángulos de conmutación de cada módulo corresponden con los creados en MATLAB.

Las primeras pruebas se han realizado con la memoria externa de la FPGA, ya que la FPGA tenía un código en la memoria interna que funcionaba correctamente, y no se quería cambiar hasta que el mío funcionase. Por eso, se utiliza el programa ADEPT, para ir mandando el programa del PC a la FPGA, hasta que el programa funcione correctamente, y poder mandarlo a la memoria interna de la FPGA.

Lo que me ha resultado más complicado, ha sido, el conseguir sincronizar la señal "syncro", con los instantes de conmutación, por lo que se ha tenido que ir cambiando el código, hasta conseguir sincronizar ambas señales. Se puede observar en la figura 6.1, que se produce la sincronización correctamente:

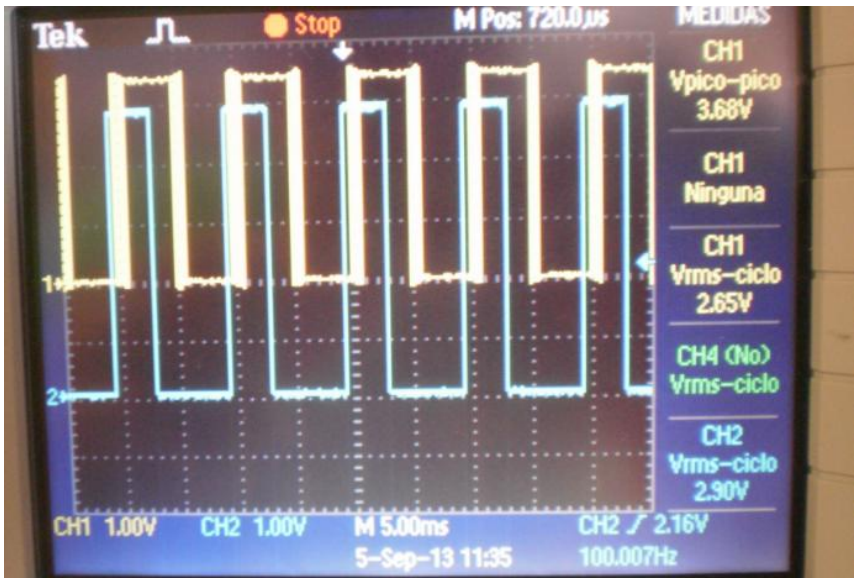


Figura 6.1: Sincronización de la señal "syncro" con los instantes de conmutación del módulo 1.

En la Figura 6.1, se observa la señal azul, es la señal de salida de nuestra placa ("syncro") .En todos los ciclos de la señal "syncro", se observa como mantiene la sincronización con los instantes de conmutación del módulo 1(señal amarilla).Esta sincronización, se produce con todos los módulos.

Una vez comprobado que la sincronización funciona, se comprueba que los ángulos de conmutación de cada módulo, obtenidos en MATLAB, de forma teórica, corresponde con lo obtenido en las salidas de la FPGA.

Se van comparando los resultados experimentales, con los resultados teóricos:

→MODULO 1: Se muestran los ángulos de conmutación del módulo 1, realizados de forma experimental y teórica, para comprobar que ambas coinciden.

-Resultado experimental:

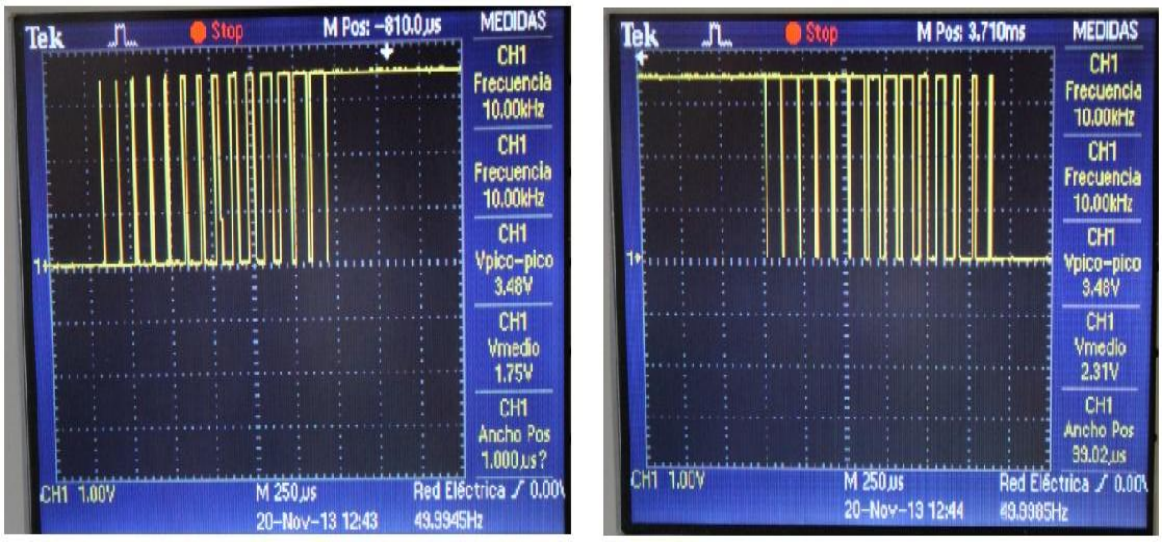


Figura 6.2: Resultados experimentales de la salida del PIN81 de la FPGA→MODULO 1

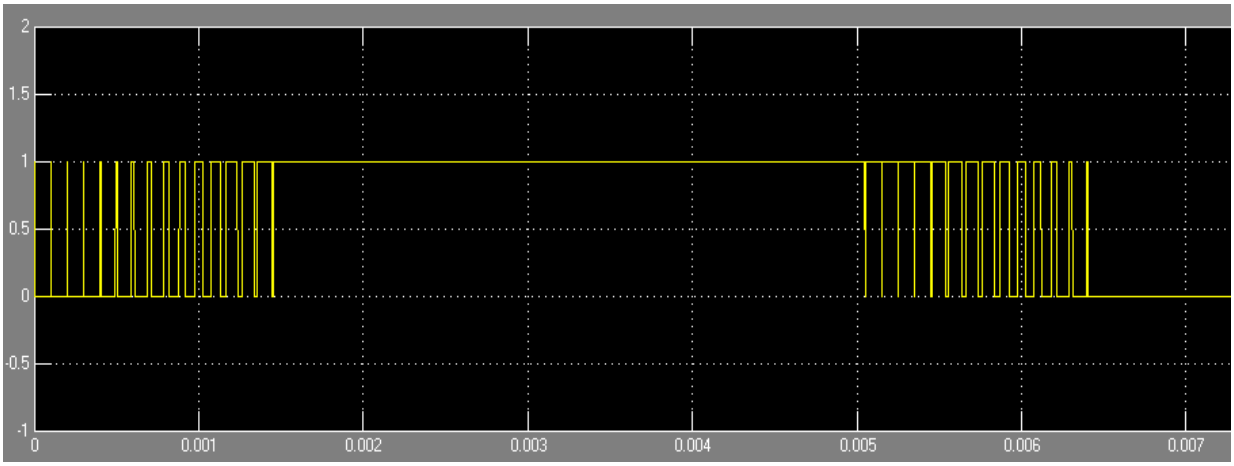


Figura 6.3: Resultado teórico del encendido y apagado de los led del MODULO 1

Se observa, que coinciden el número de flancos de subida y de bajada de cada módulo, que corresponden con el parpadeo de los led. También se mantiene encendido el módulo 1 después del primer parpadeo y apagado después del segundo parpadeo. El periodo, donde se han calculado los instantes de conmutación, es de 10(ms). Debido al zoom, para ver si coinciden los flancos, no se aprecia el periodo de la señal. Hay que fijarse en la figura 6.1, para confirmar que el periodo dura 10(ms).

→MODULO 2: Se muestran los ángulos de conmutación del módulo 2, realizados de forma experimental y teórica, para comprobar que ambas coinciden.

-Resultado experimental

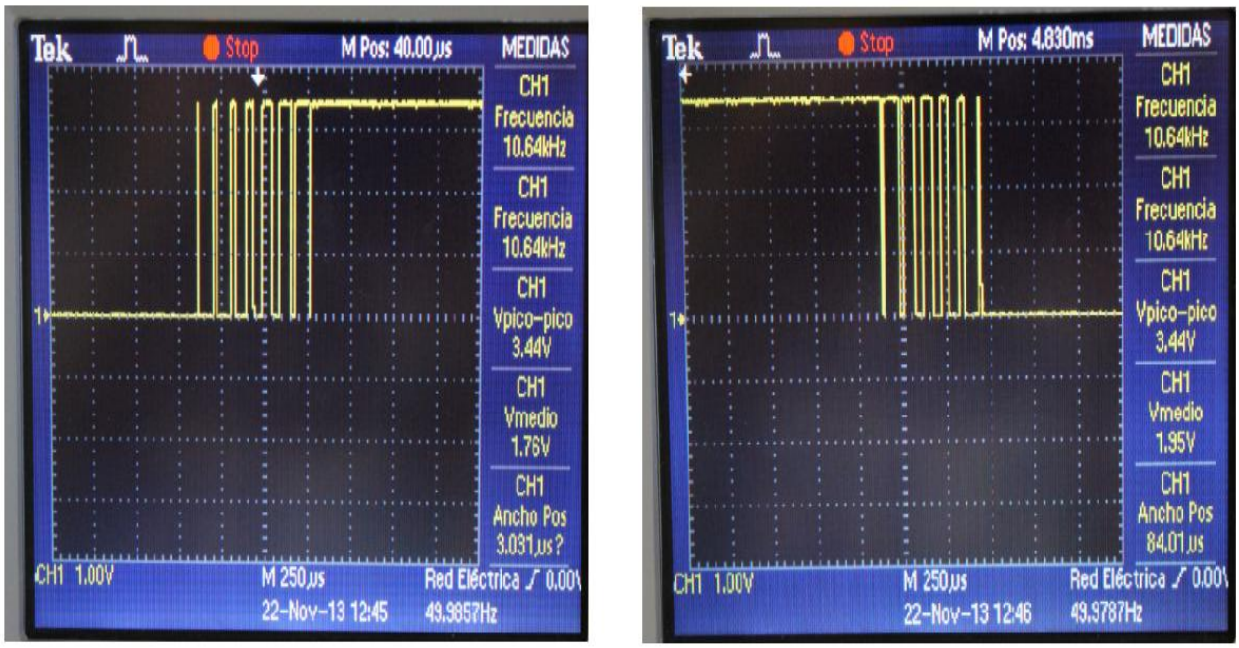


Figura 6.4: Resultados experimentales de la salida del PIN82 de la FPGA→MODULO 2

-Resultado teórico

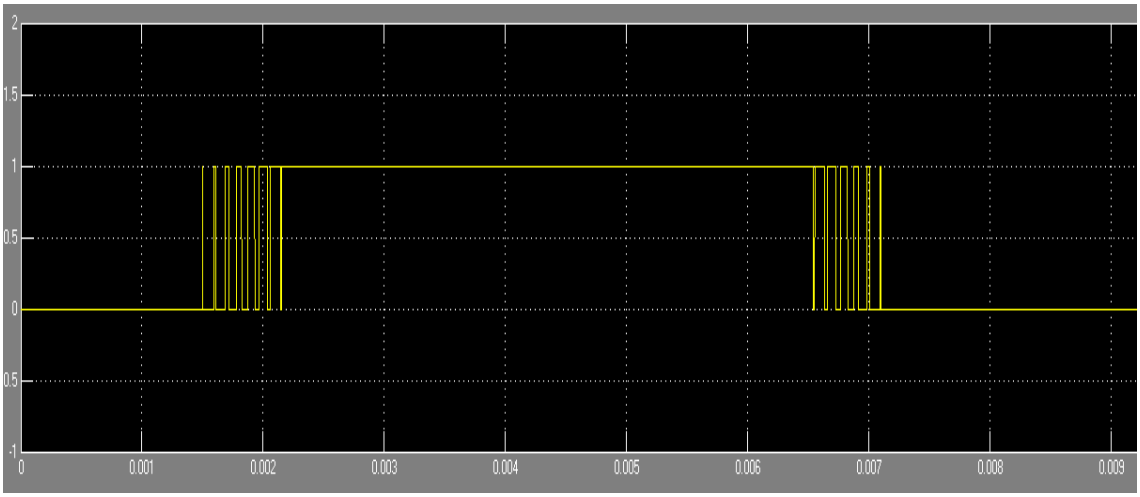


Figura 6.5: Resultado teórico del encendido y apagado de los led del MODULO 2

→MODULO 3: Se muestran los ángulos de conmutación del módulo 3, realizados de forma experimental y teórica, para comprobar que ambas coinciden.

-Resultado experimental:

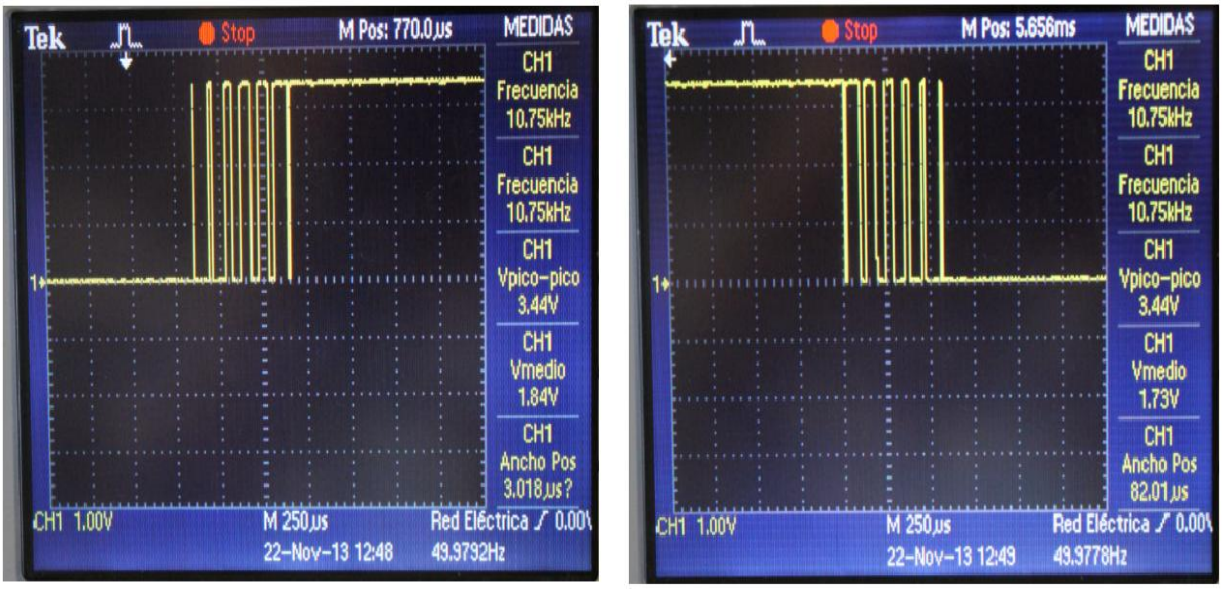


Figura 6.6: Resultados experimentales de la salida del PIN87 de la FPGA→MODULO 3

-Resultado teórico:

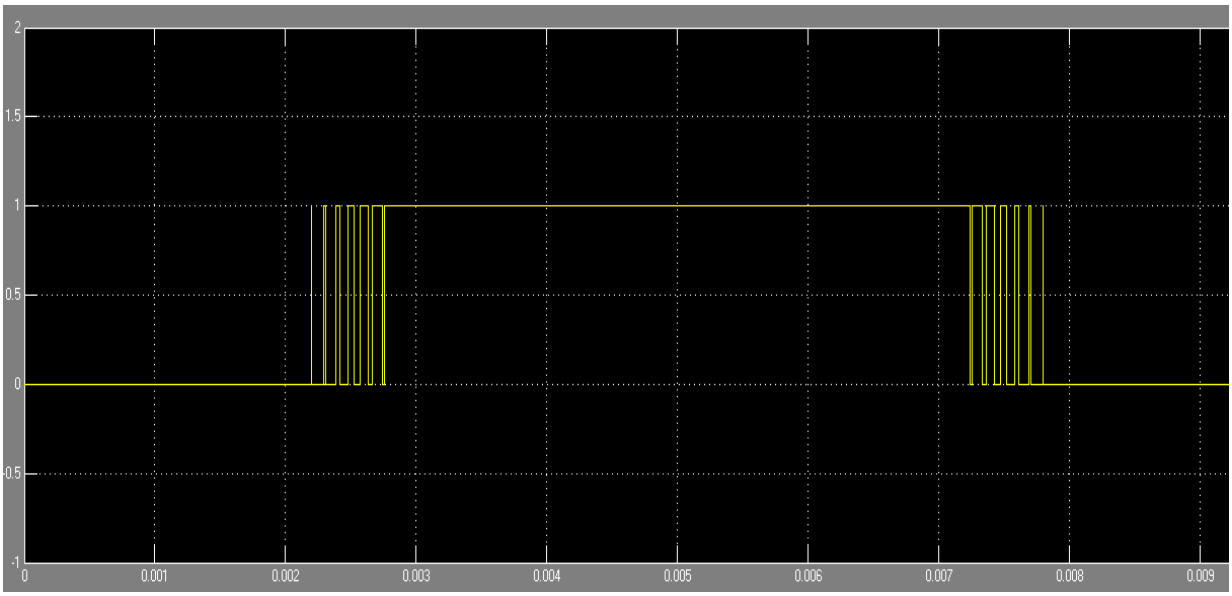


Figura 6.7: Resultado teórico del encendido y apagado de los led del MODULO 3

→MODULO 4: Se muestran los ángulos de conmutación del módulo 4, realizados de forma experimental y teórica, para comprobar que ambas coinciden.

-Resultado experimental:

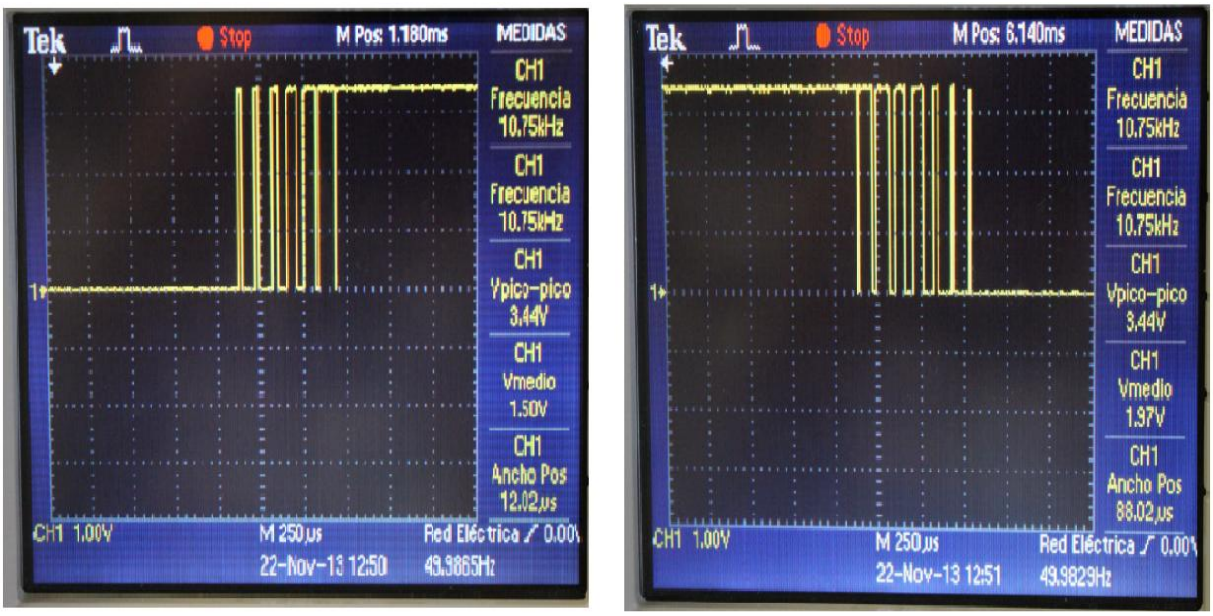


Figura 6.8: Resultados experimentales de la salida del PIN93 de la FPGA→MODULO 4

-Resultado teórico:

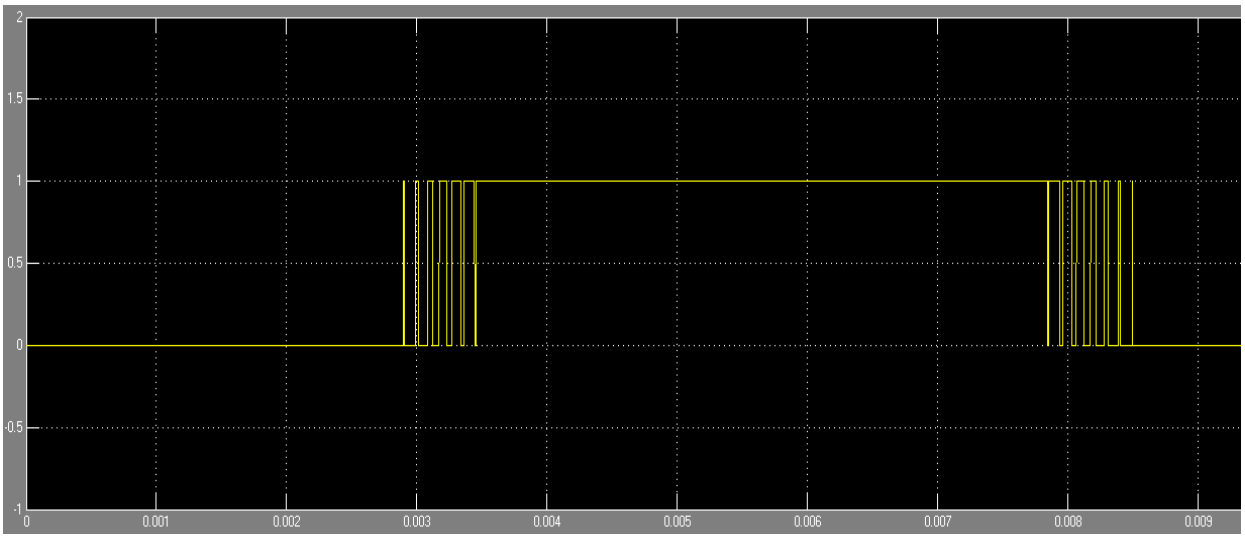


Figura 6.9: Resultado teórico del encendido y apagado de los led del MODULO 4

→MODULO 5: Se muestran los ángulos de conmutación del módulo 5, realizados de forma experimental y teórica, para comprobar que ambas coinciden.

Resultado experimental:

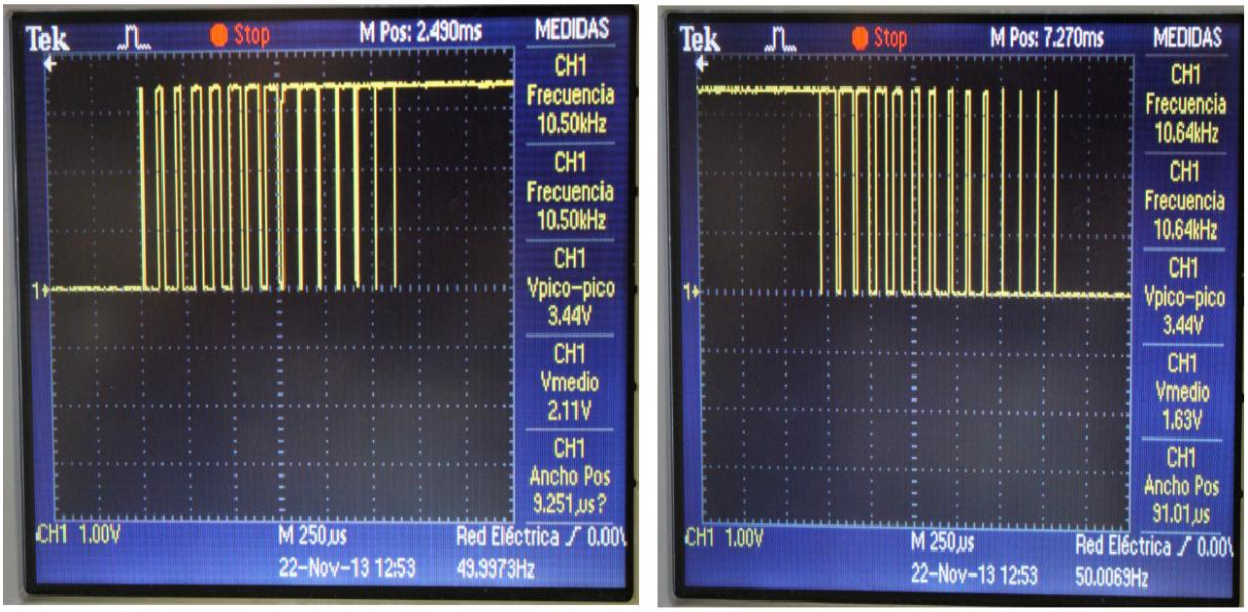


Figura 6.10: Resultados experimentales de la salida del PIN88 de la FPGA→MODULO 5

Resultado teórico:

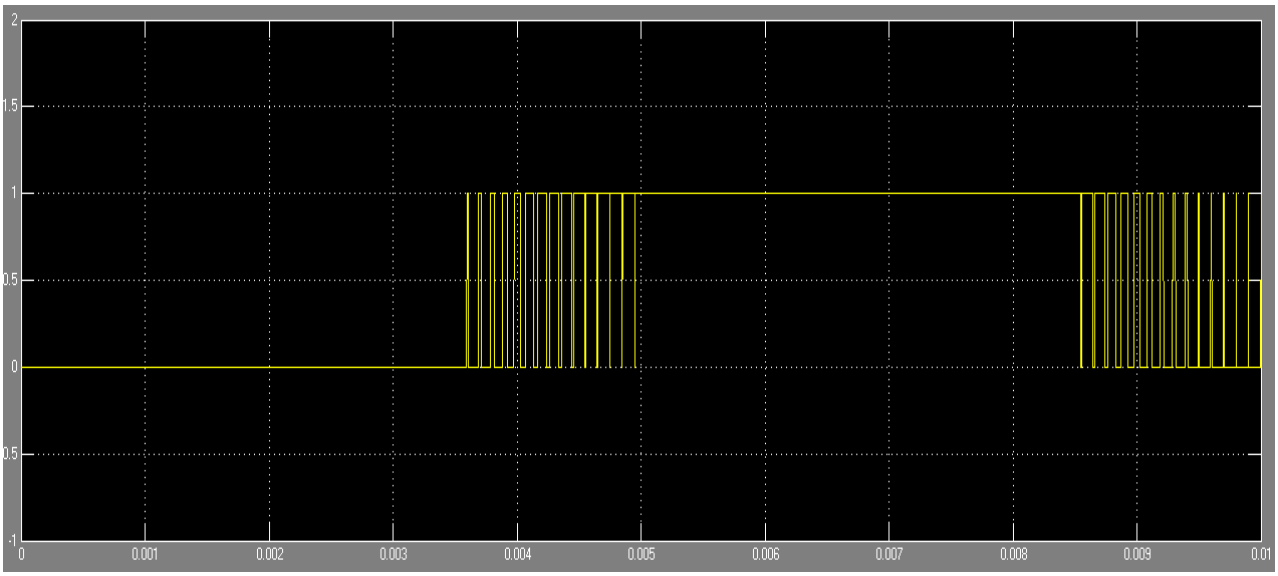


Figura 6.11: Resultado teórico del encendido y apagado de los led del MODULO 5

Se puedes ir observando como corresponden en cada uno de los módulos, los diferentes pulsos tanto en los resultados teóricos como en los experimentales.

NOTA: Después de la tensión de salida de la fpga de cada módulo, cada señal de salida pasara por un convertidor CC—CC reductor, antes de pasar a los led. Este convertidor es cogido de un proyecto anterior. Se trata de un convertidor reductor, con control de corriente, este control

se lleva a cabo mediante el componente Hv9910B, como se puede observar en la figura 6.12 y 6.13:

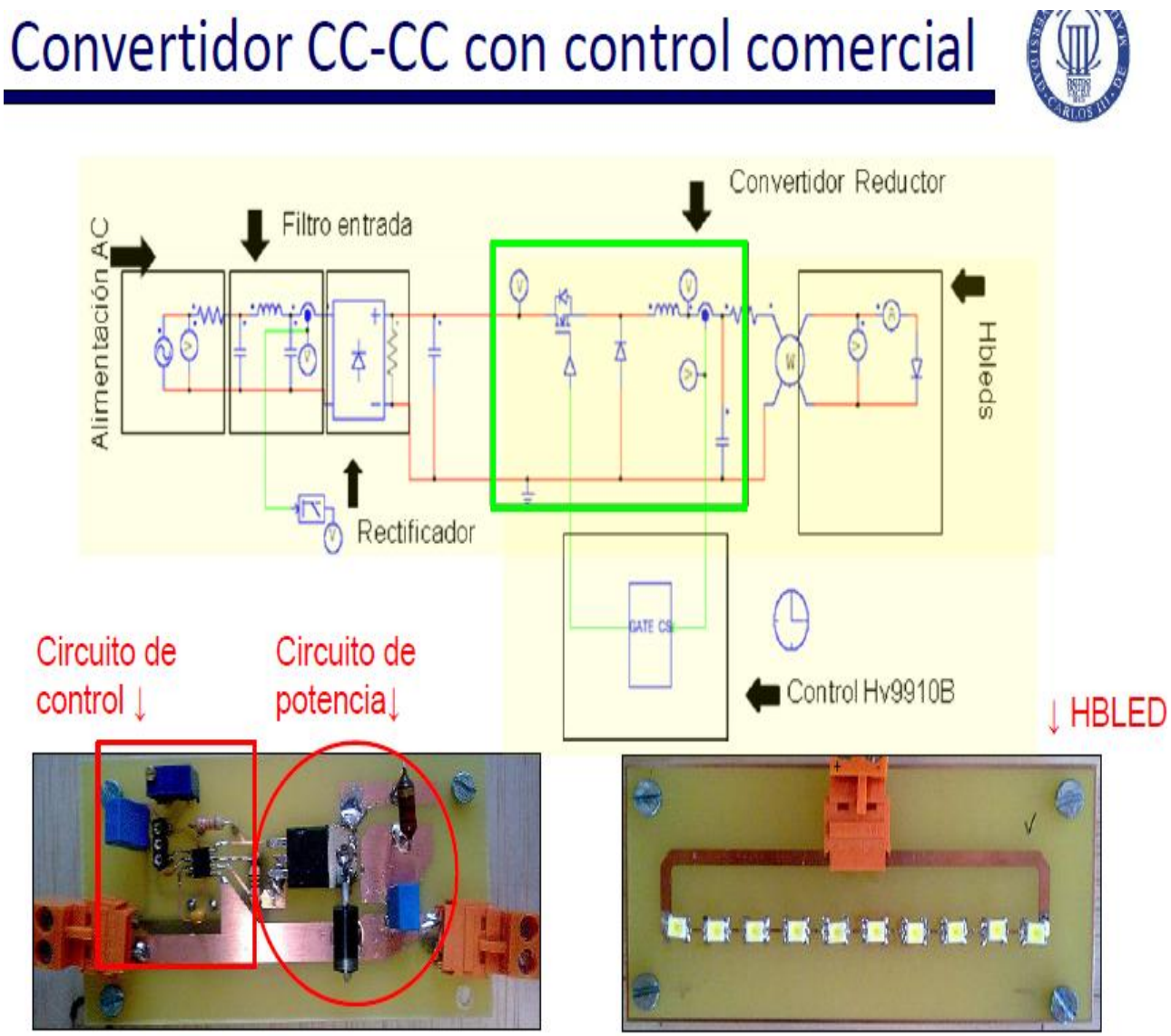


Figura 6.12 Circuito eléctrico de la incorporación del convertidor-reductor ***(Extraído del Proyecto Fin de Carrera " Diseño, construcción y validación experimental de un sistema de iluminación modular basado en LED de alto brillo (HBLED)", Andrea Toribio, 2010, Universidad Carlos III de Madrid)***

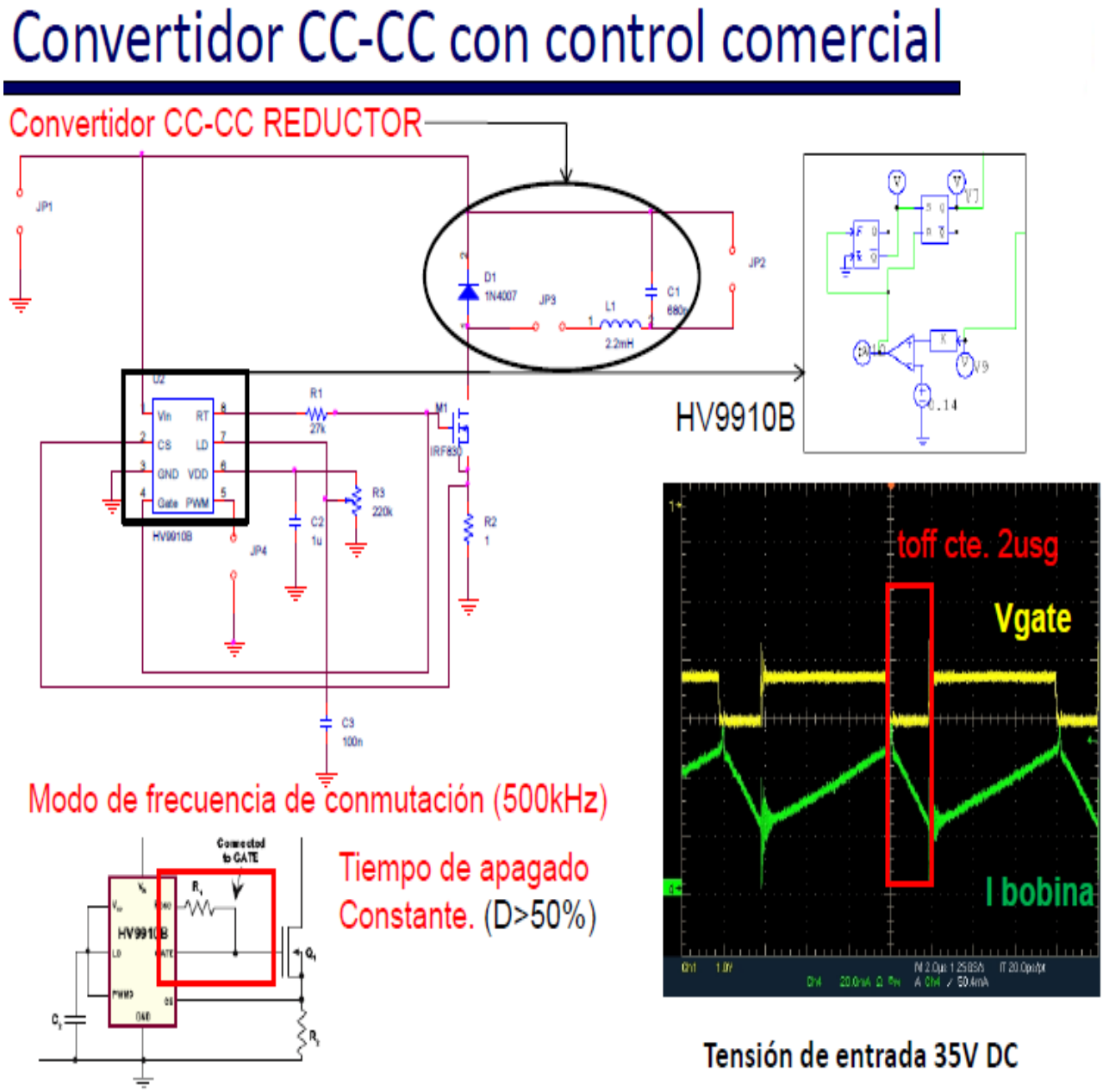


Figura 6.13: Circuito eléctrico del convertidor-reductor ***Diseño, construcción y validación experimental de un sistema de iluminación modular basado en LED de alto brillo (HBLED)***

El funcionamiento de este controlador, está configurado de la siguiente forma, el MOSFET se dispara automáticamente, cuando la corriente que pasa por el MOSFET, llega al nivel de referencia, el MOSFET se apaga durante un tiempo constante de 2µs. Así, se consigue que la corriente media que pasa por la bobina, sea más o menos constante.

Por lo que cada vez que se active un convertidor-reductor, debido al flanco de subida de la señal de salida de la fpga de cada módulo, la corriente que circulara por los módulos de los led (siempre que estén activos) será constante.

Ya con todo montado, y sabiendo que todas las partes “funcionan” correctamente, nos disponemos a probarlo todo junto, con una fuente de alterna más grande. Antes de probarlo, se introducen unos condensadores a la entrada de alterna de nuestra placa, como ya se explicó en el apartado 4, con el fin de eliminar la posible tensión de offset de la nueva fuente, ya que nos podía originar algún problema en el circuito .Se puede observar en la figura 6.14, el proyecto entero montado y en funcionamiento:

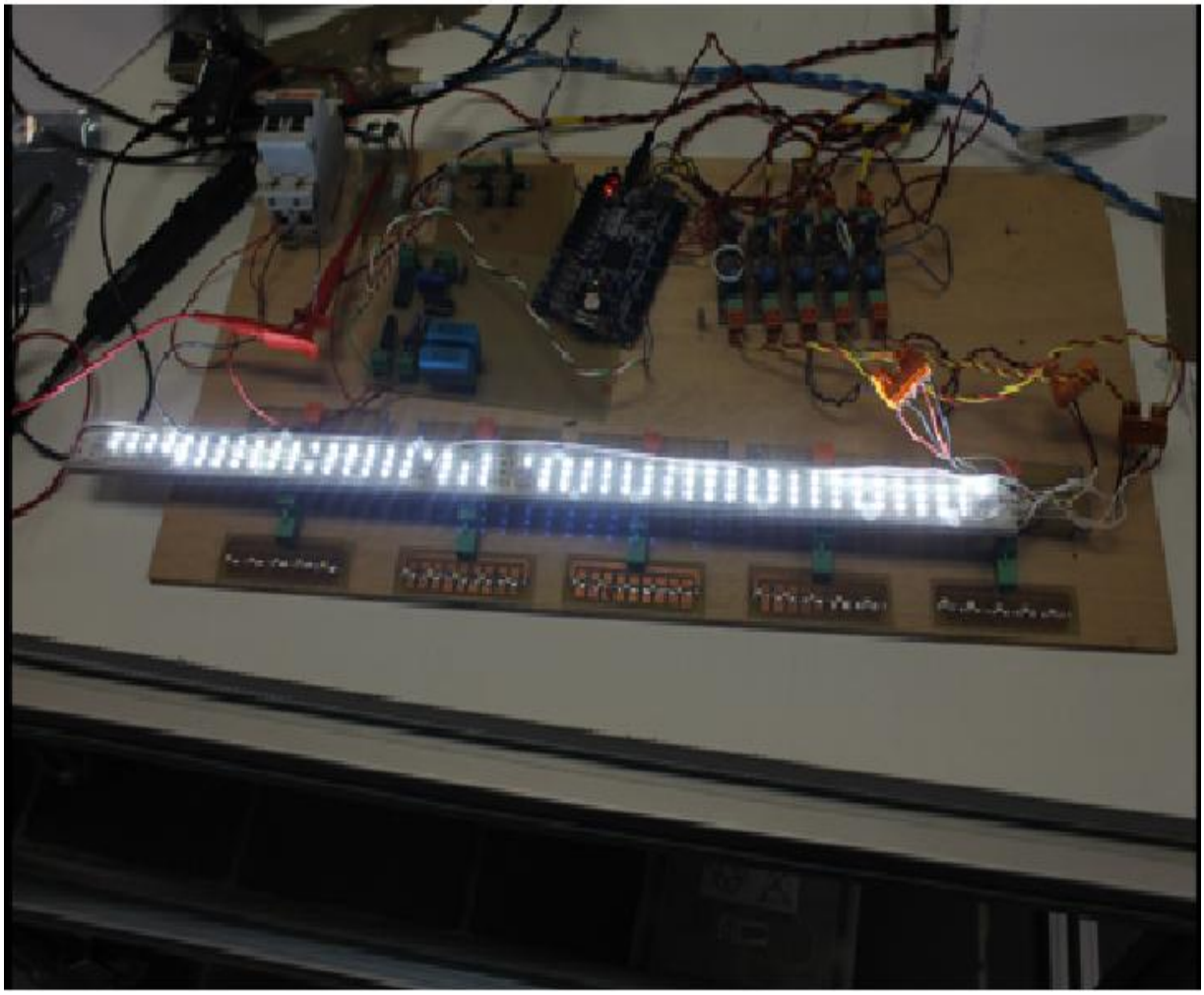


Figura 6.14: Proyecto en funcionamiento

Se observa en la figura 6.14, que hay algunos led apagados, que fueron quitados, para la separación y la correspondiente alimentación de cada módulo, como se explica en el capítulo 2.

Posteriormente se va a analizar, el comportamiento de la corriente de red, para 5 módulos, con varias tensiones de entrada. Y ver, si se cumple nuestro objetivo, la onda sinusoidal de corriente, sincronizada con la señal de red.

Vamos a representar en gráficas, la tensión y la corriente de red de los 5 módulos, para observar el resultado final.

Tensión eficaz de entrada de 80(v):

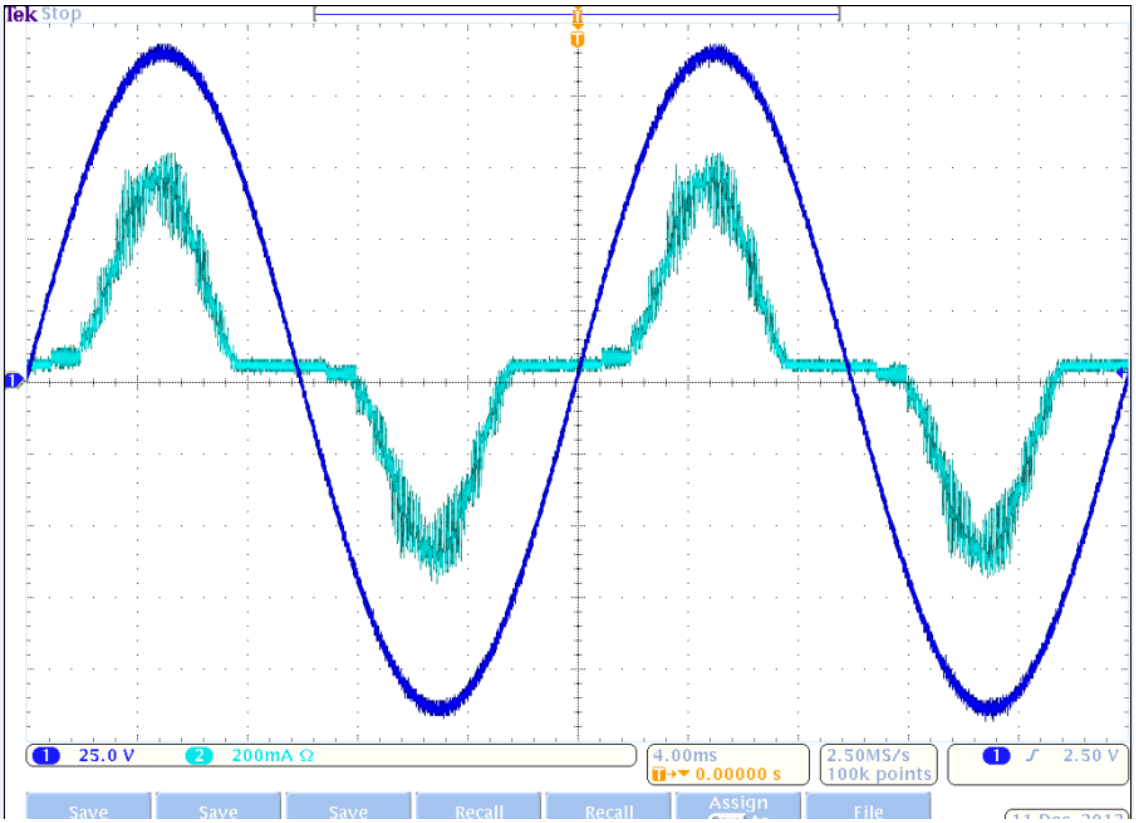


Figura 6.15: Corriente de los módulos para una tensión eficaz de entrada de 80(v)

La tensión representada en el osciloscopio son: canal 1 (azul oscuro) tensión de red, canal 2 (azul claro) corriente de red.

La corriente de red se parece a la que se quería conseguir, emulando una onda sinusoidal y diferenciándose los saltos de las corrientes de cada módulo. Tiene cierta distorsión debido a que la tensión de entrada al convertidor-reductor no es lo suficientemente grande como para que se diferencien los saltos de las corrientes de cada módulo, para valores bajos de la tensión de red.

Se puede comprobar que la señal de red y la señal de las corrientes están en fase Posteriormente se verá que al aumentar la tensión ambas señales se desfasaran.

Tensión eficaz de entrada de 140(v):

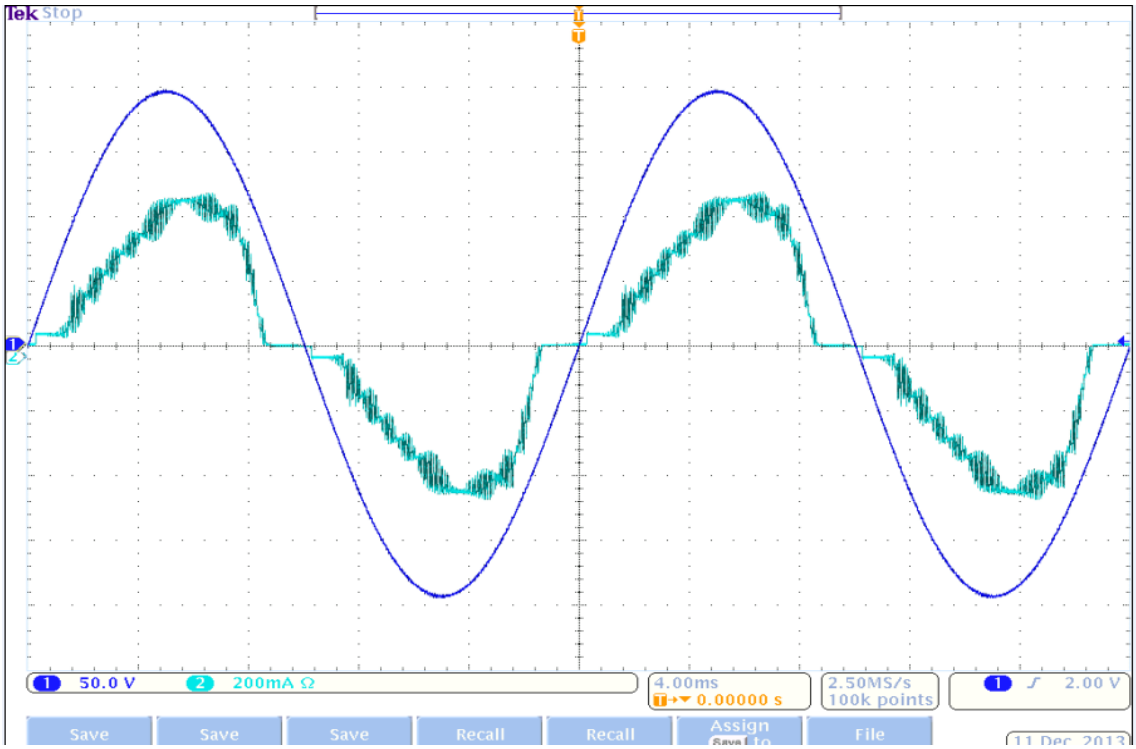


Figura 6.16: Corriente de los módulos para una tensión de entrada de 140(v)

La tensión representada en el osciloscopio son: canal 1 (azul oscuro) tensión de red, canal 2 (azul claro) corriente de red.

En cuanto a la corriente de red representada en el osciloscopio, se observa que en el primer semiperiodo se diferencia mejor los saltos correspondientes a las corrientes que pasaran por los led de cada módulo, que en la figura anterior. Esto se debe a que la tensión de red es mayor, y al convertidor-reductor le llega la tensión necesaria, para generar las corrientes necesarias, para que se distingan los saltos.

Sin embargo, en la segunda mitad de cada semiperiodo se observa que prácticamente no se ven los saltos de las corrientes de cada módulo, debido a que la onda de corriente está desfasada, respecto a la onda de la señal de red.

De los dos ensayos mostrados se deduce que para tensiones de red pequeñas, la forma de onda de la corriente es simétrica y no está desfasada respecto de la tensión de red, aunque tiene cierta distorsión debido a la deformación de la onda en los pasos por cero. Sin embargo, cuando la tensión de red tiene valores mayores, la corriente de red no tiene forma simétrica. En este caso se distinguen claramente el encendido de los módulos pero no el apagado. Esa asimetría puede deberse a un mal funcionamiento del circuito de sincronización cuando la tensión de entrada tiene un valor elevado, aunque para tensiones bajas el funcionamiento de dicho circuito se puede considerar adecuado.

7 CONCLUSIONES

Se ha conseguido el objetivo del proyecto: a partir de una señal de tensión de red, se consigue reproducir una onda de corriente alterna, similar a una sinusoidal, con varios niveles, correspondientes a los 5 módulos HBLED utilizados. Aunque la corriente tiene cierta distorsión se han mejorado los saltos de conexión de cada módulo en la onda de corriente de entrada mediante la modulación PWM, produciéndose dichos saltos de una forma más suave. La corriente de salida en cada uno de los módulos es constante durante el tiempo en el que están encendidos.

Se puede comprobar que los resultados teóricos y los resultados experimentales coinciden prácticamente con lo que se quería llevar a cabo, ya que se puede ver como en cada módulo, coinciden el número de pulsos en cada parpadeo, tanto en los teóricos como en los prácticos, y que están sincronizados con la red.

La secuencia de funcionamiento es la siguiente: cuando se produce el flanco de subida de nuestra señal de sincronización, se pone en marcha una secuencia de pulsos de ancho variable para que la corriente absorbida de la red evolucione de forma progresiva, hasta que el módulo 1 queda encendido. El módulo 1 se mantendrá encendido a la espera del siguiente parpadeo (secuencia de pulsos de ancho variable), que tendrá lugar en el segundo semiperiodo de la señal de red. Después de este segundo parpadeo, se mantendrá apagado hasta que se produzca, un nuevo flanco de subida de la señal de sincronización.

Este proceso se repetirá para los demás módulos, lo único que variara será el número de pulsos del parpadeo (es debido a la señal portadora, ya que cada pulso del parpadeo depende de las veces que la señal portadora sea menor que la onda sinusoidal) y el momento en el que se produzca ese parpadeo.

El orden de conexión y desconexión de los módulos es el siguiente:

1º: Conexión MODULO 1 **6º**: Desconexión MODULO 1

2º: Conexión MODULO 2 **7º**: Desconexión MODULO 2

3º: Conexión MODULO 3 **8º**: Desconexión MODULO 3

4º: Conexión MODULO 4 **9º**: Desconexión MODULO 4

5º: Conexión MODULO 5 **10º**: Desconexión MODULO 5

Las señales de salida de la fpga, son tensiones que corresponden con el parpadeo de cada módulo, por lo que antes de mandar, estas señales a los led, hay que pasar por un convertidor CC-CC reductor, de control de corriente (realizado en un proyecto anterior).

Gracias a la cadena de parpadeo y a las corrientes que genera el convertidor-reductor, se produce, una onda sinusoidal de corriente, en el que se puede ir observando el salto que se produce de un módulo a otro.

El parpadeo de los led y el periodo que cada módulo permanece apagado, no va a ser visible a simple vista, debido a la frecuencia de 50(Hz) de la red.

La idea realizada en este proyecto, se puede aplicar a otra combinación de módulos, es decir, se puede ampliar o disminuir el número de módulos.

8 PRESUPUESTO

En este apartado, podemos observar el desglose del coste de las partes de nuestro proyecto. El precio de los componentes será el ofertado por la empresa distribuidora de componentes electrónicos ACTRON.

Hemos dividido el presupuesto en 2 tablas, una en la que están el coste de los materiales y otra tabla las horas que he dedicado para la realización del proyecto.

-TABLA DEL COSTE DE LOS MATERIALES:

NOMBRE	CANTIDAD	PRECIO UD	PRECIO TOTAL
UAF42	1	18 €	18 €
74HC14	1	0,13€	0,13€
D1N4007	4	0,01 €	0,04 €
D1N4148	1	0,01 €	0,01 €
POT(2M)	2	0,30 €	0,60 €
CONDENSADOR(1u)	2	0,22 €	0,44 €
NMA0515SC	1	12,03 €	12,03 €
BORNAS	3	0,50 €	1,50 €
ZOCALOS	5	0,14 €	0,70 €
RESISTENCIA	9	0,01 €	0,09 €
FPGA	1	145,62 €	145,62 €
PLACA	1	7,78 €	7,78 €
TOTAL			186,94 €

Figura 8.1: Tabla coste de materiales

A continuación marcamos las referencias de cada componente:

```
                                    DIV :EUR
CANTIDAD REFERENCIA                  PRECIO    TOTAL        PLAZO
-------- ------------------------ ---------- ------------ -------------------

       1 1N4148              RoHS     0,00678         0,01 STOCK S. VENTA
           DIODO RECTI.ULTRA RAPIDO 0,15A 75V DO35         CARRETE- 10000,000

       1 1N4007              RoHS     0,00800         0,01 STOCK S. VENTA
           DIODO RECTIFICADOR 1A 1000V DO41 AXIAL          CARRETE- 5000,000

       1 100nF 63V R-5 5%    RoHS     0,02800         0,03 STOCK S. VENTA
           CONDENSADOR POLYESTER RADIAL                    CARRETE- 2800,000

       1 UAF42AP             RoHS    18,00000        18,00 STOCK S. VENTA
           CIRCUITO INTEGRADO DIL14
```

```
1 FC1016/CT17          RoHS      7,78000          7,78 STOCK S. VENTA
    PLACA C.I.PERFOR.PUNTOS 2,54 100X160 F.V

1 1uF 400V R-22,5      C/Pb      0,22000          0,22 STOCK S. VENTA
    CONDENSADOR POLYESTER RADIAL                     UNIDAD- 100,000

1 3296Y-205I-LF 2M     RoHS      0,30000          0,30 STOCK S. VENTA
    POTEN.MULTIV.AJ.VERTI.Y 9,53X10,03X4,83          BARRA- 50,000

1 BCEK254V02P          RoHS      0,50000          0,50 STOCK S. VENTA
    BORNA 2,54 2V C.I.APILABLE

1 110-87-314-41-001    RoHS      0,14000          0,14 STOCK S. VENTA
    ZOCALO C.I.2,54 14P P/RED.ORO SELECTIVO          BARRA- 34,000

1 NMA0515SC            RoHS     12,03000         12,03 STOCK S. VENTA
    CONVERTIDOR DC/DC E-5V S- +-15V 1W A-1KV         BARRA- 25,000

1 74HC14N              RoHS      0,12800          0,13 STOCK S. VENTA
    CIRCUITO INTEGRADO DIL14                         BARRA- 25,000
```

-TABLA DE EQUIPOS UTILIZADOS

DESCRIPCION	COSTE	% USO DEDICADO PROYECTO	DEDICACION (MESES)	PERIODO DE DEPRECIACION	COSTE IMPUTABLE
Ordenador	700	100	12	60	140 €
Osciloscopio	980	100	5	60	81,6 €
Soldador de estaño	200	100	3	60	10 €
Fuente de alimentacion	380	100	5	60	31,6 €
TOTAL					**263,2 €**

Figura 8.2: Tabla de equipos utilizados

-TABLA DE LAS HORAS DEDICADAS:

ACTIVIDAD	Nº HORAS	€/HORA	COSTE
DISEÑO E IMPLEMENTACION	270	32	8640 €
DOCUMENTACION	90	15	1325 €
TOTAL			**9.965 €**

Figura 8.3: Tabla de horas dedicadas

-TOTAL DEL PRESUPUESTO

TIPO DE COSTE	COSTE
Coste de materiales	186,94 €
Coste de equipos	263,2 €
Coste de ingenieria	9.965 €
TOTAL PARCIAL	10.414,14 €
I.V.A (21 %)	2.187,17 €
TOTAL	**12.601,31 €**

Figura 8.4: Total del presupuesto

9 BIBLIOGRAFIA

[1] *Estrategia de control de HBLED con reducción del condensador de almacenamiento basada en la modularización de la carga. Pablo Zumel. Grupo de Sistemas electrónicos de Potencia. Universidad Carlos III de Madrid.*

HBLED:

[2]http://www.avagotech.com AVAGO, fabricante de leds ASMT-QWBC-NHJOE (ultima vez visitada 5/2/2013).

MODULACION PWM:

[3] *Modulación de amplitud, mediante modulación por duración de pulsos (PWM o PDM) Constantino Pérez Vega. Dpto. de Ingeniería de Comunicaciones Universidad de Cantabria 2008.*

COMPONENTES:

[4] http://www.datasheetcatalog.com BURR-BROWN, fabricante del filtro UAF42 (ultima vez visitada 10/02/2014).

[5] http://www.murata-ps.com MURATA, fabricante del NMA0515SC (ultima vez visitada 24/2/2014).

[6] http://www.datasheetcatalog.com PHILIPS, fabricante del 74hc14 (ultima vez visitada 25/4/2014).

FPGA:

[7] http://www.digilentinc.com Digilent, fabricante de la FPGA BASYS.

[8] "*Digilent Basys Board Reference Manual*". Digilent Inc., 2007.

CONVERTIDOR-REDUCTOR:

[9] *Diseño, construcción y validación experimental de un sistema de iluminación modular basado en LED de alto brillo (HBLED) .Proyecto fin de carrera: Ingeniería Técnica Electrónica Industrial.2010.Andrea Toribio.*

10 ANEXO

SEGÚN ESTA EL PROGRAMA EN LA FPGA

1º) **DOCUMENTO LED.VHD**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity LED_codificacion is

Port (    Clk          : in  std_logic;
          ResN         : in  std_logic;
         modulo1     : out std_logic;
          modulo2      : out std_logic;
          modulo3      : out std_logic;
          modulo4      : out std_logic;
          modulo5      : out std_logic;
          syncro       : in std_logic;
          r1: out std_logic;--pruebas de la fpga
          r2: out std_logic
          -- r3: out std_logic;
     --synch_en: out std_logic
                );
end  LED_codificacion;

architecture Behavioral of LED_codificacion is


component  table_modulo

   port (   Clk      : in std_logic;
            ADDR       : in integer range 0 to 1023;
            DOUT       : out integer range 0 to 10000;
            LOUT       : out std_logic_vector (4 downto 0)
         );
end component;




signal tiempo          : integer range 0 to 10000:=0;
signal puntero_max     : integer range 0 to 199 :=199;
```

```
signal puntero        : integer range 0 to 199 :=0;

signal Data_IN        : integer range 0 to 10000:=0;
signal Nivel_IN       : std_logic_vector (4 downto 0);
signal Nivel_IN_aux   : std_logic_vector (4 downto 0);
signal Data_IN_tabla  : integer range 0 to 10000:=0;
signal Nivel_IN_tabla : std_logic_vector(4 downto 0);
signal ADDR_tabla     : integer range 0 to 100000 :=0;

signal tpoen          : std_logic;
signal cont           : integer range 0 to 49:=0;--contador
para acoplarlo al clk de la fpga
signal s1             :std_logic;--señal retrasada un ciclo
de reloj de la señal syncro

signal en             :std_logic;--aviso de fin de cuenta
signal synch_en       :std_logic;--pulso generada para
resetear el programa


begin

-- ****************************************
--      Instanciacion de las tablas
-- ****************************************
Tabla: table_modulo

   port map ( Clk      => Clk,
                ADDR     => ADDR_tabla,
               DOUT     => Data_IN_tabla,
               LOUT     => Nivel_IN_tabla
           );


--
************************************************************
*****************
--      Contador PRINCIPAL    --
--
************************************************************
****** --


Contador_principal: process (ResN,Clk)
 begin

  if ResN ='0' then
      tiempo<=0;
  --elsif Clk='1' and Clk'event and tpoen='1' then
 elsif synch_en='1' then
 tiempo<=0;
```

```
 en<='0';
 elsif Clk='1' and Clk'event and tpoen='1' then
      --if syncro='1' and s1='0' then
          --if synch_en='1' then
            --tiempo<=0;
            --en<='0';

          --elsif tiempo<10000 then
   if tiempo<10000 then
          tiempo<=tiempo+1;
                en<='0';
      else
          tiempo <=0;
          en<='1';

          end if;
  r1<=not syncro;
  r2<=s1;
  --r3<=syncro;
  end if;

end process Contador_principal;

enable_principal: process (ResN,Clk)
begin

    if ResN='0' then
        cont<=0;
    elsif Clk='1' and Clk'event then
      if cont =49 then
           cont<=0;
           tpoen<='1';
       else
          cont<=cont+1;
          tpoen<='0';
      end if;
    end if;

end process enable_principal;




--
************************************************************
******************--
--      Lectura  TABLA    --
--
************************************************************
****** --
```

```
lectura_tabla: process (Clk,ResN)
begin
    if ResN='0' then
        Data_IN<=0;
        Nivel_IN<="00000";
    elsif Clk='1' and Clk'event then
        Data_IN<=Data_IN_tabla;
        Nivel_In<=Nivel_In_tabla;
    end if;
end process lectura_tabla;

avance_puntero: process (Clk,ResN)
begin
      if ResN='0' then
          puntero<=0;
          elsif tiempo=0 then
                puntero<=0;
      elsif Clk='1' and Clk'event and tpoen='1' then
          if Data_IN=tiempo then
                    Nivel_IN_aux<=Nivel_In;
              if puntero=puntero_max then
                puntero<=0;
              else puntero<=puntero+1;

              end if;
          end if;
      end if;
  end process avance_puntero;

-------------sincronizacion------------
sincronizacion:  process(Resn,clk)

begin

      if ResN='0' then
          s1<='1';
      elsif Clk='1' and Clk'event then

          s1<=syncro;



      end if;
--synch_en<=syncro and (not s1) ;
end process sincronizacion;

synch_en<=syncro and (not s1) ;
```

```
--
************************************************************
*********************  VISUALIZZACION DE
MODULOS*****************--

modulo_principal: process (ResN,Clk)
begin

    if ResN='0' then
        modulo1<='0';
        modulo2<='0';
        modulo3<='0';
        modulo4<='0';
        modulo5<='0';

    elsif Clk='1' and Clk'event then
      if en<='1' then
           modulo1<=Nivel_IN_aux(0);
           modulo2<=Nivel_IN_aux(1);
           modulo3<=Nivel_IN_aux(2);
           modulo4<=Nivel_IN_aux(3);
           modulo5<=Nivel_IN_aux(4);

      end if;
    end if;

end process modulo_principal;



ADDR_tabla<=puntero;



end architecture Behavioral;
```

2°)**DOCUMENTO TABLA.VHD**

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;



entity table_modulo is


Port ( Clk   : in std_logic;
ADDR  : in integer  range 0 to 1023;
DOUT  : out integer range 0 to 10000;
LOUT  : out std_logic_vector(4 downto 0)
);

end table_modulo;

architecture Behavioral of table_modulo is
type matrice_angulo is array (0 to 199) of integer range 0
to 10000;
type matrice_bit    is array (0 to 199) of
std_logic_vector(4 downto 0);

constant angulos: matrice_angulo :=(


1,
100,
101,
200,
201,
298,
303,
397,
405,
495,
507,
592,
610,
689,
713,
786,
817,
882,
921,
978,
1026,
1073,
1131,
1168,
1236,
```

1263,
1342,
1358,
1449,
1452,
1499,
1502,
1593,
1609,
1687,
1716,
1780,
1824,
1873,
1932,
1967,
2040,
2060,
2148,
2153,
2199,
2202,
2292,
2311,
2385,
2419,
2477,
2528,
2570,
2636,
2663,
2745,
2756,
2895,
2907,
2988,
3015,
3081,
3123,
3174,
3231,
3268,
3338,
3361,
3445,
3455,
3596,
3605,
3691,
3712,
3785,

3818,
3880,
3923,
3976,
4028,
4071,
4132,
4167,
4236,
4264,
4340,
4360,
4443,
4458,
4545,
4555,
4647,
4653,
4749,
4752,
4850,
4851,
4950,
4951,
5050,
5051,
5150,
5151,
5249,
5252,
5348,
5354,
5446,
5456,
5543,
5558,
5641,
5661,
5737,
5765,
5834,
5869,
5930,
5973,
6025,
6078,
6121,
6183,
6216,
6289,
6310,

6396,
6405,
6546,
6556,
6640,
6663,
6733,
6770,
6827,
6878,
6920,
6986,
7013,
7094,
7106,
7245,
7256,
7338,
7365,
7431,
7473,
7524,
7582,
7616,
7690,
7709,
7799,
7802,
7848,
7853,
7941,
7961,
8034,
8069,
8128,
8177,
8221,
8285,
8314,
8392,
8408,
8499,
8502,
8549,
8552,
8643,
8659,
8738,
8765,
8833,
8870,

```
8928,
8975,
9023,
9080,
9119,
9184,
9215,
9288,
9312,
9391,
9409,
9494,
9506,
9596,
9604,
9698,
9703,
9800,
9801,
9900,
9901,
10000
```

```
--********************************* --
);

constant nivel:matrice_bit :=(



   "00000",
"00001",
"00000",
"00001",
"00000",
"00001",
"00000",
"00001",
"00000",
"00001",
"00000",
"00001",
```

```
"00000",
"00001",
"00000",
"00001",
"00000",
"00001",
"00000",
"00001",
"00000",
"00001",
"00000",
"00001",
"00000",
"00001",
"00000",
"00001",
"00011",
"00001",
"00011",
"00001",
"00011",
"00001",
"00011",
"00001",
"00011",
"00001",
"00011",
"00001",
"00011",
"00001",
"00011",
"00111",
"00011",
"00111",
"00011",
"00111",
"00011",
"00111",
"00011",
"00111",
"00011",
"00111",
"01111",
"00111",
"01111",
"00111",
"01111",
```

```
"00111",
"01111",
"00111",
"01111",
"00111",
"01111",
"00111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"01111",
"11111",
"11110",
"11111",
"11110",
"11111",
"11110",
"11111",
"11110",
"11111",
"11110",
"11111",
"11110",
"11111",
"11110",
"11111",
```

```
"11110",
"11111",
"11110",
"11111",
"11110",
"11111",
"11110",
"11111",
"11110",
"11111",
"11110",
"11111",
"11110",
"11111",
"11110",
"11100",
"11110",
"11100",
"11110",
"11100",
"11110",
"11100",
"11110",
"11100",
"11110",
"11100",
"11110",
"11100",
"11000",
"11100",
"11000",
"11100",
"11000",
"11100",
"11000",
"11100",
"11000",
"11100",
"11000",
"11100",
"11000",
"10000",
"11000",
"10000",
"11000",
"10000",
"11000",
"10000",
"11000",
"10000",
"11000",
```

```
"10000",
"11000",
"10000",
"11000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000",
"00000",
"10000"


--********************************* --
);


begin
Process(Clk)
begin
if Clk'event and Clk='1' then
DOUT <= angulos(ADDR);
LOUT <= nivel(ADDR);
end if;
end process;

end Behavioral;
```

3°)**DOCUMENTO SIMU.VHD**

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY simula IS
END simula;

ARCHITECTURE behavior OF simula IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT LED_codificacion
    PORT(
         Clk : IN  std_logic;
         ResN : IN  std_logic;
         modulo1 : OUT  std_logic;
         modulo2 : OUT  std_logic;
         modulo3 : OUT  std_logic;
         modulo4 : OUT  std_logic;
         modulo5 : OUT  std_logic;
         syncro : IN  std_logic;
         synch_en:OUT std_logic
        );
    END COMPONENT;


   --Inputs
   signal Clk : std_logic := '0';
   signal ResN : std_logic := '1';
   signal syncro : std_logic := '0';

     --Outputs
   signal modulo1 : std_logic;
   signal modulo2 : std_logic;
   signal modulo3 : std_logic;
   signal modulo4 : std_logic;
   signal modulo5 : std_logic;
   signal synch_en : std_logic;
   constant clockperiod :time:= 20 ns;
BEGIN

     -- Instantiate the Unit Under Test (UUT)
   uut: LED_codificacion PORT MAP (
          Clk => Clk,
          ResN => ResN,
```

```
          modulo1 => modulo1,
          modulo2 => modulo2,
          modulo3 => modulo3,
          modulo4 => modulo4,
          modulo5 => modulo5,
          syncro => syncro,
          synch_en => synch_en
        );

   -- No clocks detected in port list. Replace <clock> below
with
   -- appropriate port name



   clockprocess :process
   begin
           Clk <= '0';
           wait for clockperiod/2;
           Clk <= '1';
           wait for clockperiod/2;
   end process;

   syncprocess :process
   begin
           syncro <= '0';
           wait for 5 ms;
           syncro <= '1';
           wait for 5 ms;
   end process;
ResNprocess :process
   begin
           ResN <= '0';
           wait for 0.1 ms;
           ResN <= '1';
           wait for 80 ms;
   end process;

   -- Stimulus process
   --stim_proc: process
   --begin
      -- hold reset state for 100ms.
     -- wait for 100ms;

      --wait for <clock>_period*10;

      -- insert stimulus here
```

```
      --wait;
   --end process;

END;
```